About Component Integration Laboratories

OpenDoc is presented and maintained through an organization devoted to promoting cross-platform standards, architectures, and protocols in a vendor-independent fashion. This organization, Component Integration Laboratories (CI Labs), is composed of a number of platform and application vendors with a common interest in solving OpenDoc issues and promoting interoperability.

CI Labs supports several levels of participation through different membership categories. If you are interested in shaping the future direction of component software, or if you simply need to be kept abreast of the latest developments, you can become a member. For an information packet, send your mailing address to:

Component Integration Laboratories
PO Box 61747
Synnyvale, CA 94088-1747

Telephone: 408-864-0300
FAX: 408-864-0380
Internet: cilabs@cilabs.org

About This Book

The OpenDoc** for OS/2* Human Interface Guidelines describe the human interface mechanisms and interactions essential to the OpenDoc software architecture. It combines a discussion of the philosophy of the OpenDoc Human Interface with the actual implementation of interface elements. In addition to text-based discussions, this document makes use of screen captures and user scenarios to further facilitate the developer's understanding of OpenDoc.

Who Should Read This Book

This book is intended for people who design and develop products for use on computers running the OS/2 OpenDoc architecture.

It provides developers with background information that can help them plan and make decisions about their product design.

This document is written with the assumption that the developer is familiar with the concepts and terminology commonly used with today's graphical user interfaces and CUA* architecture. Refer to the *Object-Oriented Interface Design-IBM* Common User Access Guidelines* for information about the CUA architecture.

How This Book is Organized

This book contains seven chapters and four appendices. Developers can read these guidelines from start to finish, or use them as a reference in which to look up specific pieces of information. The following information describes the type of information that can be found in each section of the document.

- Overview of OpenDoc

  The chapter describes the basic philosophy of the OS/2 OpenDoc interface and how users and developers can benefit from this new approach.

- Part Basics

  Parts are the fundamental elements in OpenDoc that allow users to complete their tasks. This chapter describes the various ways users can represent parts. Developers should especially be aware of the discussion of the components needed for end users to be able to both edit and view these parts.

- Part Structure

  This chapter describes the basic set of menu choices available in any OpenDoc window, the new specification for OS/2 window appearance, interaction, and pop-up menus.

- Properties

Each part has a set of properties. This chapter contains a description of part properties, both those that may be modified by the user and those that may not. Properties for a particular part include kind specific preferences such as default font, and general part preferences such as the ability for a particular part to restrict the types of parts that may be embedded within it.

- Interactions

  Parts can be manipulated in several different ways. This chapter first describes how users can view parts and how they can change a part representation to an icon or a frame or to open a part into its own window. This section then describes selection and activation of parts-the essential interactions for editing both the part and its content. Next, the section describes how users can - through either menu commands or their drag-and-drop equivalents - edit parts. Finally, the section discusses how users may create and delete parts.

- Sources for Parts

  This chapter describes how users can create and access new parts through the use of templates and parts.

- Drafts, Linking, and Scripting

  OpenDoc supports several features that could greatly enhance users' productivity. This chapter describes Drafts, Linking, and Scripting. Drafts allow users to record the history of changes of a document. Linking allows users to automatically update the same information found on different pages or documents. Scripting supports the automation of tasks through easy to create scripts. Developers should read this section to determine how their products might take advantage of these features.

- Link Graphic Details

  Appendix A shows the pixel-by-pixel illustrations of the various frame states used in OpenDoc.

- Active and Selected Frame Emphasis States

  Appendix B shows the frame emphasis states.

- Crop Indicators

  Appendix C shows crop indicators.

- Pointer Graphics

  Appendix D shows the various pointers used in OpenDoc.

- Notices

  This section provides a listing of the special notices and trademarks that apply to this manual.

Prerequisite Knowledge

This document is written with the assumption that the developer is familiar with the concepts and terminology commonly used with today's today's graphical user interfaces, particularly the *Object-Oriented Interface Design-IBM Common User Access Guidelines* (CUA '91).

Related Publications

The following related publications might be useful when creating a user-interface using OpenDoc for OS/2:

- *Object-Oriented Interface Design IBM Common User Access Guidelines*
- *OpenDoc for OS/2 Programming Guide*
- *OpenDoc for OS/2 Programming Reference*
- *Open Scripting Architecture for OS/2 Programming Guide and Reference*
- *Open Scripting Architecture for OS/2 Quick Reference*

Conventions Used in This Book

The following document conventions are used in this book.

Documentation Conventions

Throughout this library of documents, the following conventions distinguish the different elements of text:

bold	Menu choices, names of keys, push buttons, check boxes, radio buttons, and field names.
italics	Titles of documents, and first occurrences of words appearing in the Glossary.
CAPITALS	File names and error message text.
Abbreviations	Mouse button 1 and mouse button 2 are designated by MB1 and MB2 after the first occurrence of each.

Overview of OpenDoc

OpenDoc is designed to facilitate the construction of compound, customizable, collaborative, and cross-platform documents. To do this, OpenDoc provides an object-oriented user model rather than an application-centric model. That is, the user focuses on a document rather than the application that manipulates that document. An OpenDoc document can contain data as diverse as navigable movies, sounds, and animation, as well as database access such as networked calendars, traditional spreadsheets, graphics, and text.

Comparison to Today's Applications

At first glance, most users will not notice the difference between an OpenDoc document and one of today's documents. Interaction techniques learned in the application-oriented world allows users to use the basic functions of OpenDoc. However, on closer inspection, the user will realize that OpenDoc documents allow much more flexibility than is available today. For example, when users paste something into a document today, they often cannot easily modify the content. Often the user must go back and modify the original document and then repaste the changes back into the target document. OpenDoc documents are much more flexible because the user can edit the content in place. Thus, there is less intrusive context-switching and all parts of the document are consolidated into a single file.

A few of today's applications have built-in editors for multiple kinds of content. However, only a few kinds of content are supported and these editors often are different from the ones used when creating the original content. However, OpenDoc uses the same editor that originally created the content unless the user asks for a different one.

Users will discover new features that make OpenDoc documents easier to create and edit than today's documents. For example, with more pervasive drag and drop, more editing can be done using direct manipulation.

Developers can and will bundle together parts into units that are designed to support sets of user tasks. Over time users will learn that they will now be able to add more functionality to their applications and that they no longer have to switch applications to create multimedia documents.

User Benefits

OpenDoc is designed to provide users with benefits that are not available in today's applications interface. The benefits OpenDoc provides are:

- *Multimedia documents*

  Any type of media that is implemented as a part can be placed into an OpenDoc document. OpenDoc can handle all current and future types of media.

- *Simplicity*

  Users can put any kind of media into a document with nothing more complex than cut-and-paste or drag-and-drop.

- *Edit in place*

  Users can edit parts right where they appear in a document, without having to cut and paste between different application windows. This lets users focus on document content, not applications. And it lets them take advantage of the context provided by the surrounding document.

- *Consolidation*

  Rather than manually managing the various files that make up a document, users can let an OpenDoc document hold the pieces. As the various pieces of content are stored in a single document, exchanging documents with other people becomes easier.

- *Consistency*

  Users can apply a common set of tools in different OpenDoc documents. They can substitute (compatible) part editors for ones that came with an application. Thus, they only need to learn one way to edit each type of data, such as text, because they can use the same text editor everywhere.

- *Tailorability*

  Users can combine generic parts from various sources to address specific tasks. They can define customized parts containing predefined content and other parts and use these parts as sources for new parts. They can replace part editors with their preferred ones, as mentioned above. They can use and create scripts to customize parts.

- *Uniformity*

  OpenDoc defines a uniform human interface for embedding and manipulating all kinds of media in documents. Part developers can write parts that fit together in a consistent interface by following these guidelines and employing the modular nature of the OpenDoc code.

- *Scalability*

  The OpenDoc Human Interface addresses a wide range of users, from novice to advanced. Novices can get started immediately; for example, first-time users can create compound documents easily, without having to understand advanced operations like linking and scripting. Experts will find no ceiling to their abilities; the open-ended OpenDoc architecture ensures a growth path.

- *History*

  OpenDoc documents have a history that can be inspected. The user decides when to create a new draft, or revision, of the document, which is stored within the same file. The user can easily look at the state of a document as it was yesterday, or two months ago as need be.

- *Forgiveness*

  OpenDoc provides better support for Undo than most platforms today. In addition, OpenDoc allows the user to Undo multiple times, rather than only once.

New Features

There are many new features of the OpenDoc Human Interface that enhance the user experience. These include:

- *Transparent Applications*

  Users don't deal with editors, except in rare situations.

- *Editor Substitution*

  Easy to replace compatible editors as user preference.

- *Edit in Place*

  Allows the user to consolidate all pieces into one document and still make necessary changes without returning to each source document.

- *Drag-and-Drop*

Available everywhere.

- *Drafts*

  Documents have a draft history.

- *Superior Linking Interface*

  Allows users to automatically update the same information found on different pages or documents

- *Intelligent Paste*

  Anticipates user expectation about the result of a paste operation when the target is a different kind than the source.

- *Templates*

  While not new, we expand the use to areas other than the WorkPlace Shell.

- *Missing Editor Problem*

  When the user doesn't have a specific editor there are fallbacks to handle the problem more gracefully than today.

- *Better Undo*

  System-wide and multiple levels of Undo allow better error recovery.

- *Concurrency*

  Multiple parts within a document may be performing different tasks at once.

Part Basics

The fundamental building blocks in OpenDoc with which users accomplish tasks are known as *parts* . This chapter describes the basics of these parts:

- How parts are represented
- How components of parts provide editing and viewing capabilities

---

User Model of Parts

OpenDoc provides an object-oriented user model, where documents are objects that contain other objects, and where each object may have a distinct behavior. However, OpenDoc uses the term *parts* rather than *objects* because parts implies that each item can be a subset or superset of another item or stand alone on its own. To users, parts appear and behave consistently because OpenDoc implements the basic part behavior and provides guidelines to help developers design individual part behaviors. This supports the goal of making users feel that parts fit together naturally within a document.

OpenDoc is an ideal architecture for multimedia documents. In OpenDoc, each new kind of medium that comes along - video, sound, animation, simulation - can be presented in parts and part templates. Thus, an OpenDoc document will automatically be able to contain all kinds of media as they appear without any modification to the existing parts.

Parts are the primitives with which users accomplish tasks. They allow all kinds of content to be edited in a single document using the user's preferred editors. Parts allow functions to be combined in flexible ways enabling them to address tasks more effectively.

Part Representations

The user sees each part as a self-contained entity. It has contents, behavior, and a set of properties. Parts can contain data and other parts as well.

Parts can be represented in the following ways:

- Icons

  An icon (large, small, or thumbnail) is a small picture that stands for a part. Icons serve the same role in OpenDoc as they do in todays standard graphical user interfaces (GUIs). Icons provide handles to parts, allowing them to be manipulated as a whole. Since icons are visual representations of the part, and do not display the part's contents, a user cannot edit the contents of a part from the icon.

- Frames

  A frame not only represents a part, but also contains a view of the part's contents. Since a frame displays the part's contents, a user can usually edit the contents of a part from within a frame.

Icons and frames are alternate representations-a single instance of a part may appear as an icon or as a frame, but not both.

A part inside an OpenDoc document is often represented as a frame; in a folder window, parts are often represented as icons. When represented as a frame, a part's content is usually visible and editable. However there are exceptions. For example, a sound part may display a button or icon instead of a visible representation of the sound itself. Clicking on the button might play its sound, not edit it. A database-query part may display the values being returned, perhaps as a graph, instead of the query definition, which might be a property of the part.

It is important to understand that users do not manipulate parts directly; they manipulate frames and icons. Frames represent the part and the part's content, and icons represent the part to the user. A part's content may be seen in several different frames at once. For example, a table part may be seen as a chart in one frame and as a table in a different frame. A container part may have a mixture of frames and icons at once. Any frame may be viewed as an icon at any time, without affecting any other frame of the same part.

In addition, a part be represented in multiple frames. When the user deletes a frame or icon, the corresponding part is not necessarily deleted. For example, if a part is represented in both a chart frame and a table frame and the user deletes the chart frame, the table frame is not deleted. Therefore, the user's data is not deleted.(**)

Properties are characteristics about a part. Each part has several properties; for example, whether it should be displayed as an icon or a frame, which editor to use, who last modified the contents of the part and when.

Part Icons

An icon is a small picture that represents a part. Icons serve the same role in OpenDoc as they do in today's standard graphical user interfaces (GUIs). They provide handles to parts, allowing them to be manipulated as a whole. Icons may be dragged on the Desktop and between windows. They may be the target for drops during drag operations. They may be opened into windows and closed back into icons. They may also be changed into frames and changed back into icons.

An icon will have a different appearance when a window is open on the part it represents; for example, OS/2 adds cross-hatching to the background of an in-use icon.

The standard OpenDoc representations are: large icon, small icon, thumbnail, and frame. Part developers are encouraged to support and implement these representations for their parts. Standard views in OpenDoc for OS/2 are: Icons, Details, Tree, and document.

Icon Size

Part icons in OS/2 have three possible sizes (in VGA mode):

- Large icon (32 x 32 pixels)

  It has the same size and general appearance as other OS/2 icons. When OpenDoc parts are represented as icons, there is nothing unique about their appearance that would distinguish them from WorkPlace Shell objects.

- Small icon (16 x 16 pixels)

  This is the size used when icons are viewed as small icons in folders, or in Details views.

- Thumbnail (64 x 64 pixels)

  This is a new size introduced by OpenDoc. It differs from the other sizes in two respects:

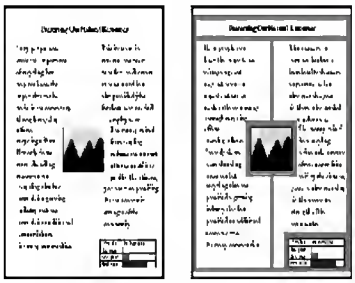
  - It is larger than the other sizes
  - Instead of showing a generic shape for all instances of a type of part, it shows a miniature representation (thumbnail) of a page of the part's contents.

Part Frames

A frame is an area of the display screen that represents a part. Frames are normally rectangular, but they need not be. Like icons, frames:

- Provide a handle onto parts, allowing parts to be manipulated as a whole
- Can be dragged on the desktop and between windows
- Can be the target for drops during drag operations
- Can be opened into windows and closed back into frames
- Can be closed into icons and reopened back into frames

Unlike an icon, however, a frame allows a part's contents to be seen and edited.

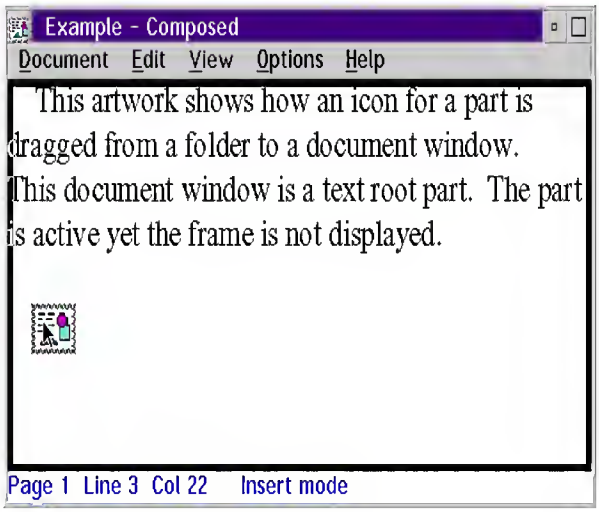


Frames showing how a document is composed of embedded parts

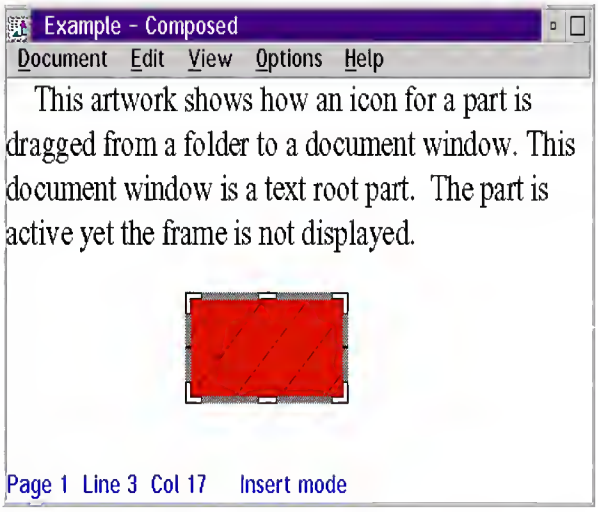
Frames allow a part's content to be edited in place, without requiring the user to open separate windows. In addition, there may be many frames showing the content of a single part. For example, when editing a 3D drawing the user may wish to see a top view, side view, face-on view, and wire-frame view. Or, a part might provide a graphical and tabular view of its contents. Or, the header on a page layout part might be showing the same part on each page. Any change the user makes in one frame is immediately reflected in all other visible frames on the same part. Because different kinds of parts have different requirements for passing changes to all frames, OpenDoc provides the capability but does not specify the interface.

Frames (like icons) can be opened into windows. Such windows are transitory views, not permanent representations of parts, while frames are persistent. When a frame is opened into a window, the frame remains visible in its place. Because frames already show the contents of parts, one might ask why one would open a frame into a window. The answer is that some parts contain more content than can be displayed in their frames. For example, a spreadsheet part may be quite large, with only a portion of it appearing in a particular frame. Opening it into a window allows the entire spreadsheet to be seen and edited and allows the user to adjust which which portion will appear in the frame. See Displaying Parts for more information.

The preferred representation of contained parts is a property of the container. Some parts, such as folders, prefer to show embedded parts as icons. Others, such as text documents, prefer to show them as frames. When a part is copied from a Desktop folder to an OpenDoc document, the copied part changes its representation from an icon to a frame (see An icon for a part is dragged from a folder to a document window.). Conversely, when a part is copied from an OpenDoc document to a Desktop folder, the copied part changes its representation from a frame to an icon. In neither case do the contents or properties of the copied part change. But once inside a container, part's representations may be specified on a part by part basis.



An icon for a part is dragged from a folder to a document window



The icon becomes a frame in the document window

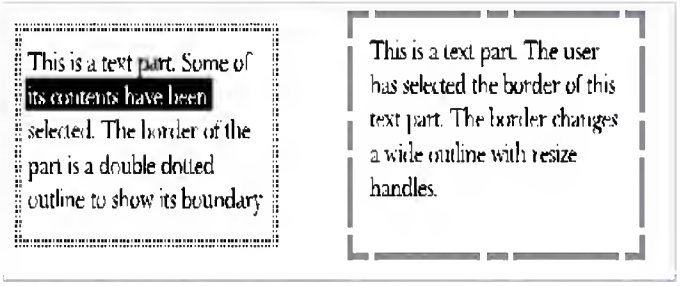
Frame States and Borders

This section refers only to parts in a frame representation. A frame may be in one of three states: *active* , *inactive* , or *selected* . A frame is active when it contains the selection (which may be null); this state is indicated by the active frame border, as shown in Two different frame states.. Normally, the active part receives commands and keyboard events, and its menu, palettes, and other user-interface elements are displayed. The user-interface elements of the previously active frame are hidden. Only one frame at a time may be active.

In general, whenever the user clicks mouse button 1 (MB1) in a part frame, the frame becomes active (if not already active). Large icon, small icon, and thumbnail representations do not become active. They become selected and their container becomes active. If the part does not support activation, then the part becomes selected and its container is activated. Parts that do not support activation are those that have been bundled or those that are provided as simple parts.(**)

Being inactive does not mean that a part isn't running. Parts may execute asynchronously whether they are active or inactive, even if they are displayed as icons.

In addition to activating frames, the user may select them. Selecting frames is described in Selecting Parts. Selected frames appear differently from active frames. The active frame has a standard appearance but the selected frame appearance is determined by the container as shown in the following figure.

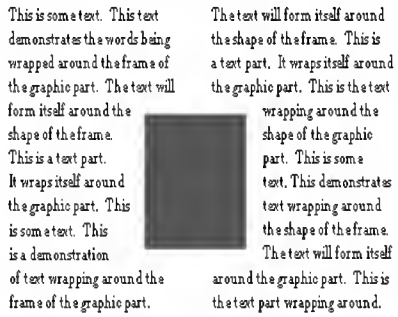


Two different frame states
Note that when a frame is selected as a whole, it is not active. Its container is active. Every part in OpenDoc, except root parts are contained in another part.

An active graphics frame containing a selected text frame

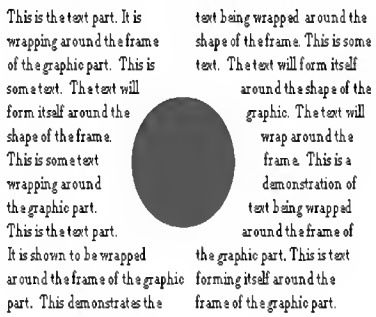
Frame Shape

A frame's shape is the area available for displaying the frame's contents. A frame's shape is often rectangular, although it need not be. For example, a button might have rounded corners. Some applications may enable direct manipulation of the shape; others may provide indirect manipulation by a menu or Properties notebook; still others may provide no control over the shape. It is not required that the user be given control over the shape.



A rectangular frame

A frame's shape determines how closely other data may wrap to its visible contents.



An irregularly shaped frame

Bundled Frame

To treat all embedded parts inside a frame as a single atomic object, a frame can be *bundled* . Bundled frame contents cannot be selected. Any mouse click inside the frame's contents selects the entire frame. This allows the frame to be easily manipulated as a whole. This behavior of a bundled frame is not exactly the same as the effects of the **Group** command often found in graphics programs; **Group** simply ties a set of selected objects together, while bundling a frame treats all embedded parts as a single object, including the frame.

Part Contents

Parts can contain both intrinsic data and other parts. A part's intrinsic data is the contents directly in the part (not data contained within the part's embedded parts). This data is called the part's *intrinsic content* . For example, the intrinsic content of a text part would be zero or more characters. The intrinsic content of a graphics part might be lines, circles, rectangles, and other shapes, while the intrinsic content of a video part is digitized video. Likewise, sound parts contain digitized sound, and simulation parts contain executable code.

In addition to its intrinsic contents, a part may contain other parts as shown in the following figure.

A text part with an embedded simple part

Part developers are encouraged to allow their parts to contain other parts. However, part developers may or may not choose to allow other parts to be placed inside their parts. A key characteristic of OpenDoc is that if a part can contain one kind of part, it can contain all kinds of parts, since OpenDoc supplies a uniform wrapper for parts. (Contrast this with the small number of standard data types supported today, such as text, PICT and TIFF). To a part, other parts are opaque black boxes. Parts know nothing about the internal structure or semantics of other parts. A part allowing other parts to be embedded in it just means that the part is capable of handling the basic part wrapper.

Part Structure

A part that resides on the Desktop or in a folder is considered a *root part document* . This document is accessible from the file system, for example, a document that resides in an OS/2 folder.

A root part is not embedded within another part. In OpenDoc, parts can serve as root parts or as embedded parts. The root part controls the basic layout structure of the document (text and graphics). Because the document can now contain embedded frames displaying any kind of content, the document is no longer a block of data controlled by a single application. Rather it is comprised of smaller blocks of data controlled by many software components.

The top-level part of a window is the part on which the window was opened. This part may be a root part or an embedded part (that has been opened into its own window). The top-level part establishes the basic behavior for that window, including:

- Basic editing metaphors (text scroll, drawing, spreadsheet, etc.)
- Saving paradigm (whether saving is manual or automatic, currently available only for the root part)
- Valid printing options for that part.

Editors and Viewers

The behavior of a particular kind of part is implemented by two pieces of software: an *editor* and a *viewer* . Typically, the user modifies the content of a part using its editor, and on occasion might be able to only view the part's content. For an example of a viewer, consider that most users will need only a video viewer, while a few users will have an editor that will produce or edit video documents. This section describes editors and viewers in more detail.

Editors as Applications

Part editors and viewers replace the traditional concept of applications in OpenDoc. Theoretically, anything a user can do in a current application a user could do with an OpenDoc part. For example, a user could have a spreadsheet that looked just like one of today's spreadsheet applications. However, there would be big differences in the parts compared to today's spreadsheets:

- How the part was built

  With OpenDoc, applications will typically be built by combining off-the-shelf part editors with custom built ones. Part developers can provide all the functionality the user desires without having to write all the pieces.

- How the part is accessed

  OpenDoc for OS/2 has an object-oriented user interface. The user will rarely need to deal directly with editors. The user simply opens the part and the part opens using its preferred editor or viewer.

- Ability to add new parts

  The user can purchase additional parts and their editors, such as new charts, and immediately use them with existing parts on the system.

- Ability to replace parts

  The user can replace pieces of an application that are implemented as separate part editors. For example, if a spreadsheet application included a chart maker implemented as a separate part, the user could substitute a different chart maker. This gives users far more tailorability than is available today.

Editor

An editor is a piece of software that displays a part's content and provides a user interface for modifying that kind of content. The user interface may include menus, controls, tool palettes, rulers, and other interaction techniques.

For example, a text part editor interprets the intrinsic data stored in the part, and draws the characters inside the part's frame, breaking lines of text to fit the frame's width if necessary. This editor allows the user to select ranges of text, using the mouse or other input devices, and apply menu choices (such as Bold and Italic). More sophisticated versions of a Text part editor might provide rulers with direct-manipulation tab settings and margins.

A part editor may support multiple kinds of data; for example *Surf Writer 2.0* might support Styled Text and AcmeSoft Text in addition to its own data format. Usually all the kinds supported by a common editor are in the same category. See Properties for more information about kinds and categories.

If a part is capable of containing other parts, its editor takes these contained parts into consideration when displaying its content. Imagine a text part that contains a graphics part. The text parts editor might wrap the text in such a way that the text does not run across the graphic part's boundaries. In other situations, the text may actually be overlaid, as in a document with a corporate logo in the background. Exactly how contained parts are handled with respect to intrinsic content is up to the part developer. However, the part editor is not responsible for drawing the contents of a contained part; that is the responsibility of the contained part's editor (or viewer).

The editor code is a distinct object, separated from content, thus promoting code sharing. However, the editor is not typically seen as a distinct object by the user. Unlike applications today, the part editor appears to be rolled into the part itself. The user sees the part content and editor as one single object. Whenever the user moves a part to another location, the part editor appears to moves with it.

The editor itself becomes visible to the user as an object only when the editor is being installed or replaced, or the preferred editor of a particular type of part is being set. See Installing and Removing Editors.

Viewer

When the user wants to send a document to a colleague, the user may not want to include the part's editor in the document for one of two reasons: the colleague may not have licensed the kinds of parts contained in the document, or the user might want the colleague only to read (and not modify) the document. The *viewer* comes into play in these situations.

A viewer is a subset of its corresponding editor's code that displays and prints a part's content. For example, a text part viewer contains just enough code that understand the intrinsic content and present it to the user on the screen or paper. Still, it should render the content with the same fidelity as the editor.

Because the viewer is not designed to support editing the part's content, several of the basic choices found on the menus are disabled. For example, Save and Revert to Saved, might be disabled as well as Undo, Redo, Cut, Paste, Paste as, and Delete. The part developer may choose to enable any one of these menu items, if appropriate, or include advanced features in the viewer, such as cropping or resizing the intrinsic contents.

Similarities Between Editors and Viewers

Both editors and viewers are capable of interpreting the stored data contained in the part and presenting that data to the user, perhaps dynamically (such as sounds or movies).

Editors and viewers are both pieces of software that must be installed on the user's system. Without either an editor or viewer for a particular part type, the part's data cannot be viewed or manipulated by the user.

Differences Between Editors and Viewers

Once an editor is installed on a user's system, the user does not pass the editor software to other people, unless permitted by a license agreement.

Viewers, on the other hand, may be freely distributed, to enable and encourage document interchange. However, viewers provide only a limited menu subset.

Part Structure

This chapter describes the menu structure for parts and the window behaviors. *Drafts* , a special feature of OpenDoc root part documents is described in Drafts.

Menus

The *document shell* provides the basic menu bar when a document is opened. These menus are defined below. In addition to the menu bar, each part has a pop-up menu just like other objects in the WorkPlace Shell. Pop-up menus contain only choices that are currently available for a part, while menu-bar menus show all possible choices, with those currently unavailable grayed out.

Menu-Bar Contents

When an OpenDoc document is open, several standard menu choices are displayed in the menu bar.

Note: The **{Active Part}** menus are not part of the standard menu. Parts may add these menu choices.

It is recommended that the parts add choices to the appropriate menus, according to the scope of the choice.

The standard menus are and their contents are:

- **{Top-level Part}**(**)

  Choices that affect the top-level part shown in the window. In a document window, this choice is named Document. When an embedded part is opened into a window, this first menu choice will be renamed to the category name for that part.

- **Edit**

  Standard choices that affect selected parts or selected intrinsic data.

- **View**

  Choices that affect the view of the active part in a window or choices that display other parts associated with the active part (such as tool bars and palettes).

- **Active** (optional)

  Choices that affect the currently active part in a window.

- **{Additional menus affecting the active part}**

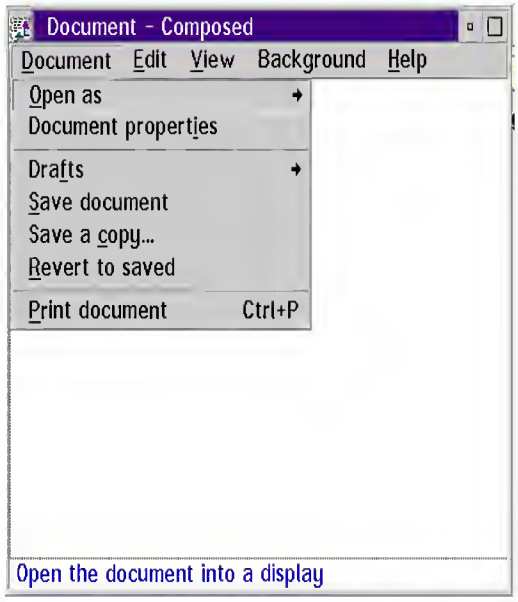
- **Help**

  Choices that provide Help on the active part.

{Top-level Part} Menu

This menu always applies to the top-level part in the window. The mnemonic used for this menu is **D** if Document is used as the name. If another category name is used, some other letter that fits the category name should be used as a mnemonic.

The base menu items: **Edit**, **View**, and **Help** are the same on the top-level and embedded-level menu's. The **Document** menu pertains only to the top-level menu. An embedded part's first menu will be specific to the part.



Document Menu

Document Menu

This menu appears in the window on a root part document. The menu choices apply to the root part.

The following choices open another window on the document:

Open as
: A cascaded menu that lists the different views available for the part. All container parts need to support the Icons, Tree, and Details(**) views in addition to their part-specific views. Choosing this choice opens another window on the document into the specified view.

Properties
: Opens a Properties notebook on the root part.

The following information describes data-transfer and convenience choices in the document:

Drafts
: A cascading choice leading to **Create draft** and **Draft history**.

Save

Saves a copy of the current document (root part) and all its embedded parts. If **Perfect Save** is available as an option to the user, **Save** will be grayed whenever **Perfect Save** is set.

Revert to Saved

Reverts to the last saved version of a document and all its embedded parts. The user cannot **Undo** this action.

Print...

Prints the entire document.

{Embedded Part} Menu

This menu appears in a window that has been opened on an embedded part. This menu always applies to an embedded part that has been opened into its own window. When an embedded part is opened into a window, the first menu choice in the new window will be named to the category name for the part.

The following information describes choices that open another window on the part.

Open as

A cascaded menu that lists the different views available for the part. All container parts need to support the Icons, Tree, and Details(**) view in addition to their part-specific views. This choice opens another window on the part.

Properties

Opens a Properties notebook on the top-level part in the window.

The following information describes data-transfer and convenience actions in the embedded part menu:

Print...

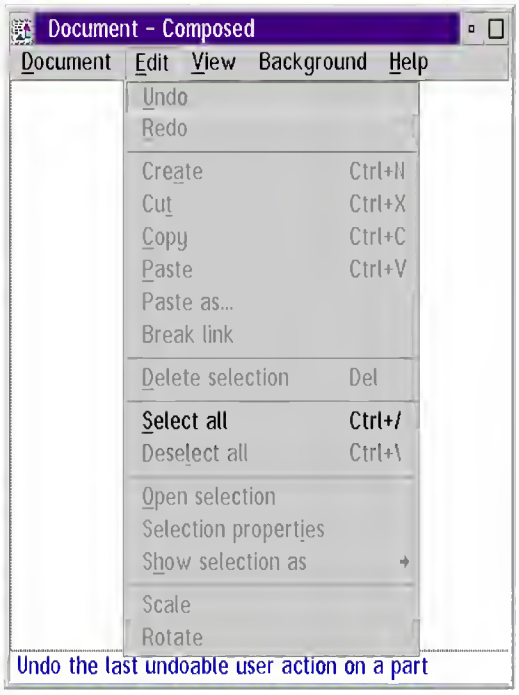
Prints the embedded part. A part adds this menu choice only if it supports printing.

Edit Menu

These choices vary in scope, as follows:

- **Undo** and **Redo** affect everything in the document.
- **Create**, **Cut**, **Copy**, **Paste**, **Paste as**, **Break link** , and **Delete** apply to the selected parts or data.
- **Select all** and **Deselect all** apply to all contents of the active part.

The mnemonic for the **Edit** menu is **E**.



Edit Menu

The **Edit** menu shows the use of **Ctrl+X**, **C**, **V**, for **Cut**, **Copy**, and **Paste**. Parts should also support the OS/2 Presentation Manager standard of **Shift+Insert** for **Paste**; **Ctrl+Insert** for **Copy**, and **Shift+Delete** for **Cut**.

Undo {*action*}

Reverses the effects of the most recent user action that has not been undone. **Undo** restores all parts to their states before that action. Not all user actions are reversible. OpenDoc supports multiple levels of **Undo**. For example, if the user selects **Undo** three times in succession, then the last three reversible actions are reversed.

Redo {*action*}

Reverses the effects of the last **Undo** action that hasn't been redone, restoring all parts to their states before the **Undo**. OpenDoc supports multiple levels of **Redo**. Each time the user selects **Redo**, the previous **Undo** action, if any, is redone. This choice is available for use only until all previous Undos have been reversed.

The following are data-transfer actions:

Cre**a**te

Creates a new part of the same type as the selected part. A new part will be created on the clipboard.

Cu**t**

Removes the selection from its current position and places it on the clipboard. When followed by a **Paste** choice, this provides the **Move** operation. See Cut, Copy, Paste, and Create.

Copy

Leaves the selection undisturbed as it places a copy of the selection on the clipboard. When used with the **Paste** choice, this provides the **Copy** operation. See Cut, Copy, Paste, and Create.

Paste

Pastes the contents of the clipboard in the cursored position or in a default position in the target part. When used with the **Cut** and **Copy** choices, this provides the **Move** and **Copy** operations. See Cut, Copy, Paste, and Create.

Paste as...

Provides the same function as the **Paste** choice, but also displays a dialog box that allows the user to specify the data format for pasting the clipboard into the destination. Converts the clipboard contents to the user-specified format. **Paste as** also allows user to create links. See Linking Documents.

Break link

(For target links only) This choice will break the link with the source. The action cannot be undone. A confirmation dialog box will be given before the link is broken. See Linking Documents.

Delete selection

Removes the selection from the active part. This action is not undoable.

The following information describes convenience actions on the **Edit** menu:

Select all

Selects all the contents of the active part.

Dese**l**ect all

Deselects all the contents of the active part.

The following information describes actions that apply only to selected parts.

Open selection

Opens the selected part into a window of its own. The part will appear in the new window in its default view. If more than one part is selected, this choice will open each selected part into a separate window.

Selection properties

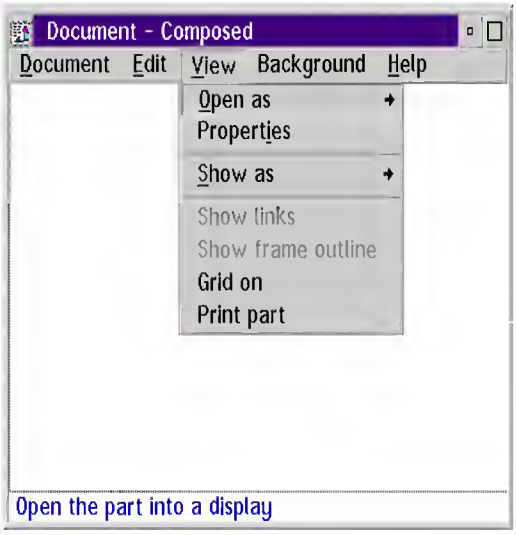
Opens a properties notebook on the selected part. If more than one part is selected, this choice will open a separate properties notebook for each selected part.

Show selection as

Changes the representation(s) of the selected embedded part(s) in the current window. A cascaded menu contains representations available for the selected part.

View Menu

Controls the views of the active part and the display of related items like tool bars or palettes. Any views the user can change should be put as the first items in the **View** menu. All container parts should support Icons, Tree, and Details views. The mnemonic for the **View** menu is **V**.



View Menu

Note the status area at the bottom of the window. This area provides information about the interface element under the cursor or pointer.

{View x}

The various views available for the active part (Table and Chart for example). If several views exist the developer can place them in a cascaded menu called **View** *part* as.

Open As

Opens the active part into a new window of its own.

Icons

All embedded parts are displayed as icons (like a folder in OS/2). Intrinsic data are not shown.

Tree

The tree of embedded parts is shown. Parts are displayed as icons in a tree. Initially only the first level of the tree is displayed, but users may open and close branches.

Details

Shows a list of parts with their small icons and information such as Name, Size, and

Creation date.

Properties
: Opens a Properties notebook on the active part.

Actions related to the display frame borders follow:

Show Frame Outline
: Appears in the View menu for embedded parts opened into a window. When invoked, an outline of the frame is shown to indicate the portion of a part's content displayed in its frame in the containing window. The user can drag this outline to adjust the visible region of the part. For more details, see Editing Frames. Toggles with **Hide Frame Outline**.

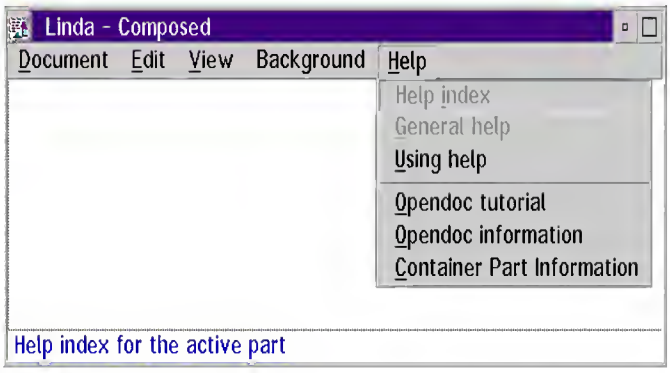
Show links
: Shows the link border for all links in the active part. Toggles with **Hide links**.

{Active Part} Menus

In OpenDoc, the active part's menu-bar choices display choices related to the currently active part. These are not a standard menus. Part developers can add these to the base menu-bar. Developers can add more than one active part menu, with names appropriate to their contents. Choices that pertain to content selected within the part should be separated from whole-part choices. Notice in the artwork that displays the pulldowns from the menu-bar, certain choices are grouped together by the action they provide on the part.
Help Menu

In OS/2, the help menu behaves as defined in the *Object-Oriented Interface Design - IBM Common User Access Guidelines* , except that in addition to the Product Information menu, there is part information that the active part should display. When a Windows** part is active the Help menu will contain the Windows Help Manager** choices.

The **Help** menu contains choices for the active part. The mnemonic for Help is **H**.



Help Menu

Help Index
: Index for the active part.

General Help
: General help for the active part.

Using Help
: Information about how to use the help facility.

Tutorial
: Tutorial for the active part. This is an optional item, the part developer needs to add this menu item within the part.

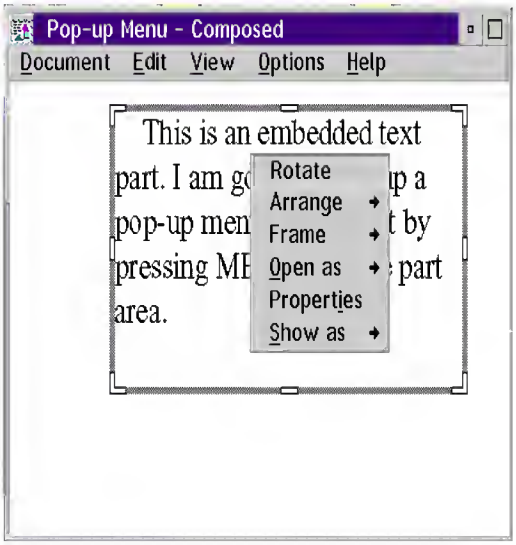
OpenDoc tutorial
: This choice leads to a tutorial about using OpenDoc.

part information
: Information on the active part. This is an optional item; the part developer needs to add this to the menu items within the part.

Pop-up Menu

A pop-up (or context) menu contains choices that are currently available for the part under the pointer. Pop-up menus are provided to give the user immediate access to the choices applicable to the content under the pointer. For intrinsic data, the choices in the pop-up menu are determined by the part handler of the selected data. The default mouse method of invoking a pop-up menu in OS/2 is to click MB2. Clicking MB2 over a part that is not selected or activated will move activation to that part and display a pop-up menu.

For actions on parts, there is a minimum set of actions that should be supported by all parts. Parts can provide additional options, but all parts should provide the following:



Pop-up Menu

The pop-up choices for parts follow:

Open as->

Cascades to a list of different views (for example, Icons, Details, Tree) available for the part.

Properties

Opens a Properties notebook on the part.

Show as->

Changes the representation of the source of the pop-up. A cascaded menu contains representations available for the part. If this setting is changed, the **Show as** setting in the part's Properties notebook is also changed since they are the same setting. Standard representations are Large icon, Small icon, Thumbnail, and Frame. The name Frame may be replaced with the names of the frame views supported. For example, the developer could have Chart and Table in the list. If there are many views you may put them into a cascaded menu called Frame.

Help

Cascaded menu for help choices (typically **Index**, **General**, **Using Help**, **<Part> information**.

Parts should add the following choices, as applicable, to the pop-up menu:

Note: This is not a complete set of choices-developers should add any choices that are appropriate to the part.

Cre**a**te

Creates a new part of the same type as the part under the pointer. A new part will be created on the clipboard.

Cu**t**

Removes the part from its current position and places it on the clipboard. When followed by a **Paste** choice, this provides the Move operation. See Cut, Copy, Paste, and Create.

Copy

Leaves the part undisturbed as it places a copy of the part on the clipboard. When used with the **Paste** choice, this provides the **Copy** operation. See Cut, Copy, Paste, and Create.

Paste

Pastes the contents of the clipboard in the cursored position or in a default position in the target part. When used with the **Cut** and **Copy** choices, this provides the Move and **Copy** operations. See Cut, Copy, Paste, and Create.

Paste As...

Provides the same function as the **Paste** choice, but also displays a dialog box that allows the user to specify the data format for pasting the clipboard contents into the destination. Converts the clipboard contents to the user-specified format. Also, allows links it be created. When the **Create** choice is selected, and the **Paste** As choice is subsequently

chosen, the **Paste link** pushbutton in the **Paste as** dialog should be grayed since linking is not possible at this point. See Linking Documents.

Break link

(For target links only) This choice will break the selected link with the source. The action cannot be undone. A confirmation dialog box will appear before the link is broken. See Linking Documents.

Delete selection

Removes the selection.

The following information describes convenience actions:

Print...

Prints the source of the pop-up.

Additional actions that can be performed on the part should also be listed in the pop-up menu. Related choices should be grouped together. If the container part's actions cannot be placed intelligently in the pop-up menu, they should appear after the part's actions and before the **Help** choice. Generally, the order of choices in the pop-up menu are:

- Choices that open new windows (for example, **Open as** and **Properties**)
- Clipboard/data transfer (for example, **Cut**, **Paste**, and **Delete**)
- Convenience choices
- Part specific actions
- Container actions (for example, **Align**)
- Help

A divider appears in the menu between each category of choices.

Pop-up Menus and Multiple Selections

If more than one part is selected and the pointer is on a part that is selected, the pop-up menu contains the choices that apply to all of the selected parts. In the first release of OpenDoc, limited support for this feature is provided. There are three different cases to consider. Case 1 is when the multiple selection includes intrinsic content only; case 2 is when multiple embedded parts are only selected; and case 3 is when both intrinsic content and embedded parts are selected.

In Case 1 the part editor should be able to provide the intersection of choices that apply to all the selected intrinsic data.

In Case 2 where multiple embedded parts are selected, at a minimum, the pop-up should provide the following actions:

Open

Opens the default view of each part.

Properties

Opens a Properties notebook for each part.

Show as

{Large icon, Small icon, Thumbnail, and Frame}

Delete

Deletes the pop-up's source, or all selected parts or content.

The developer is free to add more choices to the pop-up menu like **Align** or **Arrange** that the container can perform on the parts.

In Case 3, at a minimum any container choices (for example, **Align**) that can be performed on the selected parts should be displayed. Part editors can add more choices as long as they apply to the part.

Pop-Up Menus and WorkPlace Shell Menus

Ideally, pop-up menus for an OpenDoc object on the Desktop should contain the same actions that you would get when bringing up a pop-up menu from inside an OpenDoc part. But because the WorkPlace Shell uses a different data-transfer model than OpenDoc(**), and uses shadows instead of links, there will be some differences between the pop-up menus depending on the context in which they are displayed. For instance, OpenDoc part editors do not contribute to the WorkPlace Shell menus.

There are several different kinds of sources for pop-up menus:

- An icon in a folder on the Desktop

- An opened OpenDoc document
- Contents of an opened OpenDoc document

If the user brings up a pop-up menu from inside an OpenDoc document (for example, by clicking in the background of the root part) then the pop-up menu will contain the OpenDoc Clipboard and Link actions. Note that the **Open as** menu choice will list the available views for the part.

Window Menu not Supported

Many applications include window menus to allow the user to find the documents that are currently open within that application. Because each document is treated as a separate process in OpenDoc, this function is available by examining the Window List. Thus, a window menu is not required.

Plug-Ins and Services

Plug-ins are functions that extend the capabilities of the document shell and session-wide functionality of OpenDoc.

When a document is opened, the document shell will start all installed shell plug-ins. When called by the document shell, a plug-in can add its own dispatch or focus modules, add entries to the name spaces, or or patch session-wide objects (like clipboard).

Note: Part editor developers should implement plug-ins.

Service components are mechanisms to extend the capabilities of parts. Menu items for actions provided by these services are the responsibility of the part editor using the services and should be added to the menu bar.

OpenDoc components that provide services to parts' or documents' spell-checking or database access tools, when developed as service components, can be applied to a either a single part, all the parts of a document, or across several documents.

Window Types

In many respects, OpenDoc windows resemble OS/2 WorkPlace Shell windows. Windows are used to present views of documents or other parts. The two types of windows are parent and child windows (or primary and secondary windows).

Parent Window

A parent window is a document window opened on a root part. Because a parent window is always a document window, the menu bar will contain the document menu choices such as **Save** and **Revert to Saved**.

Child Window

A child window is a window opened on an embedded part. The child window carries its own menu bar similar to that of the parent window, but lacking some of the Document choices such as **Save** and **Revert to Saved**.

Opening and closing windows on a child part has no effect on the selection state in its original document.

When a part is opened into a window, that part becomes the active part in the window and displays its menus in the opened window. When a child-window is closed, the source part should be in the same state that it was before it was opened into a child-window, unless the user changed the selection in the window from which the part was initially opened.

If the part has a name, it will appear in the title bar of the child window, followed by its parent document name and the View name of the child window. See Child-Window Title Bar.. Thus the title will have the following format:

```
<Name> from <Document name> - <View>
```

Truncation of the title, if needed, will start from the right.

If the user closes the parent window, any child windows associated with that parent window will also close.

Child-Window Title Bar

Window List

A window list is an OS/2 feature that shows users which windows are currently open on their destop and provides access to those windows. The entries on the window list are the title bar contents of each window.

Opened OpenDoc windows appear in the window list along with the WorkPlace Shell windows. Embedded parts opened into their own windows and appear indented under their parent document windows.

Properties

OpenDoc uses an object-oriented user model, where documents are objects that contain other objects, and where each may have distinct behaviors. Each part has contents, behavior, and a set of properties.

Properties are information about a part or intrinsic content. For example, a file on the Desktop has properties such as name, kind, size, where and when it was created. The text you are reading has properties such as font, size, style, color, and spacing. A picture embedded in this document has properties such as horizontal and vertical position relative to page, margin, text, or border.

Some part properties can be modified by users, while others may be set only by developers or the system. There is a basic set of part properties that OpenDoc expects each part to support. Developers are responsible for deciding which additional properties to support. The basic set of properties is discussed in the following section.

All the properties in the parts property notebook refer to part-specific properties except the Bundled and View As properties, which refer to the current frame or icon.

Part Kinds and Categories

In this section, two widely-used part properties are defined: *kind* and *category* . These properties affect many aspects of the behavior of the part.

Part Kind

A *part kind* refers to the data format of a part's contents; for example, *AcmeSoft Text 8.0* or *SurfWriter II* . Developers initially assign part kinds.

Part Category

A category refers to a set of part kinds. A category describes the type of data contained in a part, such as text. Conceptually, a category groups part kinds that users perceive as similar. Users may think of a category as a generic term for several brand name variants. For example, the category Styled Text might include the kinds: *SurfWriter 3.0* , *SurfWriter 2.0* , and others.

Note: SurfWriter is used as an example product set.

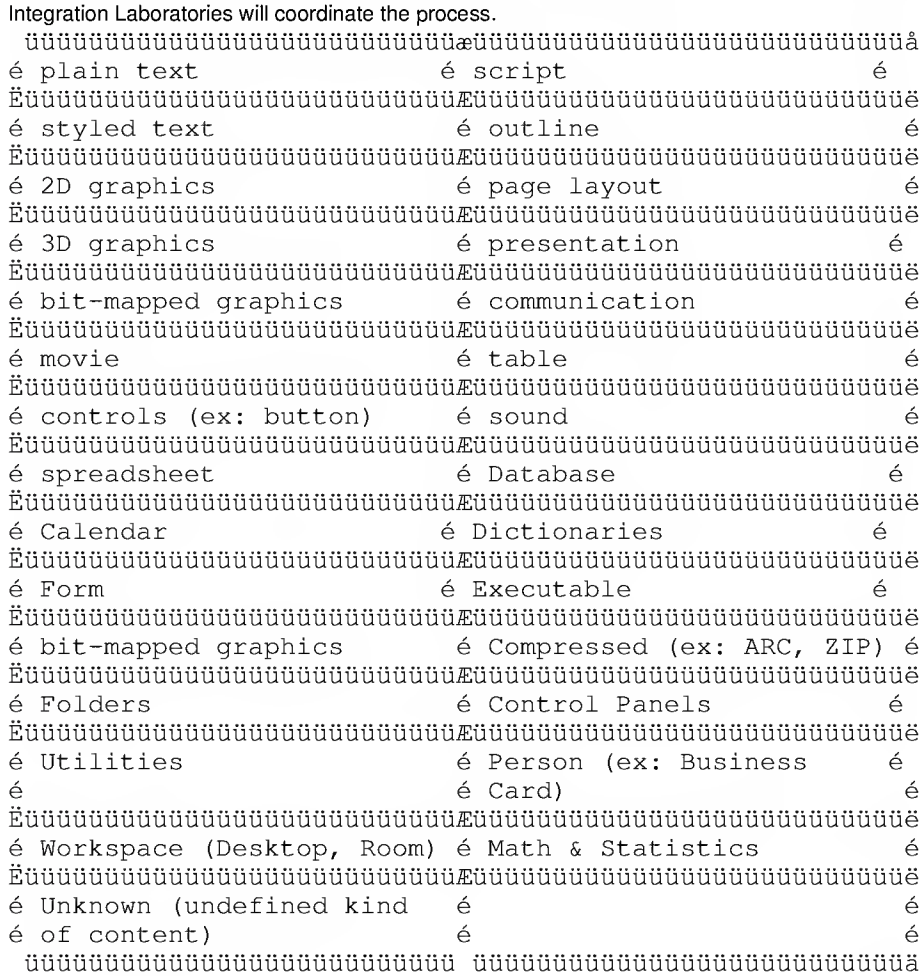
Developers assign categories and they cannot be changed by users. OpenDoc uses categories to determine the set of part editors or viewers that are applicable to a given part.

When users want to change the editor or viewer for a part, they need to see the valid alternatives. There may be hundreds of editors and viewers in the system, but the user interface displays only those that are applicable to the part. The system determines the subset to show by comparing the editor and viewer types with the part's category and filtering out the ones that do not match. See Part-Editor Properties.

Categories are not hierarchical. A category cannot include other categories. If a part can accept multiple categories of data, the categories must be listed individually in the Category property, separated by commas (for example, *Plain Text* , *Styled Text*).
Initial categories

The following initial set of categories for OpenDoc parts is suggested.

Developers are free to define additional categories. It is up to developers to agree on the categories among themselves; Component Integration Laboratories will coordinate the process.



User-modifiable Properties

These basic properties can be modified by the user:

Name
A string of text identifying the part. The name is displayed as the icon label as well as in the window title when the part

is opened into its own window. It can be used in scripts to refer to the part. The same naming rules that are used in the WorkPlace Shell will also apply in OpenDoc. The user is not required to name parts; default names can be constructed dynamically as needed, usually for scripting purposes.

Template

A Boolean check box: ON means that the part behaves as a template; OFF means that it behaves as an ordinary part.

Comment

A text string containing descriptive information about the part. This is purely for the user's benefit and has no meaning to the system.

Bundled

A Boolean check box: ON means that the frame's contents cannot be modified directly by the user; OFF means that they are modifiable. A bundled frame's contents cannot be selected or modified directly by the user. However, its contents can be changed indirectly by a script, link, or other means. The bundled property, unlike most other basic properties, applies only to the current representation of the part and not to the part itself.

Kind

A text string displaying the part's data format. For example; SurfWriter 3.2 or SurfDraw 1.9. The user can change the kind of a part by selecting an existing alternative format of data, which is displayed in a drop-down list. See Properties for more information.

Editor

A pointer to the editor that the user prefers to use on this part's content. By default, it is the last editor used on this part's content. When users inspect the editors available, they see a see a filtered list containing only those editors and viewers in the system that can handle the part's kind. The user can select either a viewer or an editor from the list to replace the one associated with the part. If the preferred editor is not available at some point, the system-wide editor properties control substitution.

Show as

A one-of-many property (large icon, small icon, thumbnail, frame(**)) that specifies the preferred representation of displaying the part. The developer can add other names of methods.

Non-Modifiable Properties

Users can not modify these properties directly.

Category

A text string displaying the part's category. A category is a general classification of a part's contents. It describes the type of data contained in the part (for example, *Text*). The property can contain more than one category separated by commas; for example *Text* , *Styled Text* . See Part Kinds and Categories for more information.

Created

The date and time the part was created. The time is not reset by **Copy** operations; the copy has the same creation date as the original.

Modified

The date and time the part was last modified.

By

A text string identifying the user who caused the last modification. This is optional.

Size

A number representing the number of bytes required to store the part in the file system.

Properties Developers Define

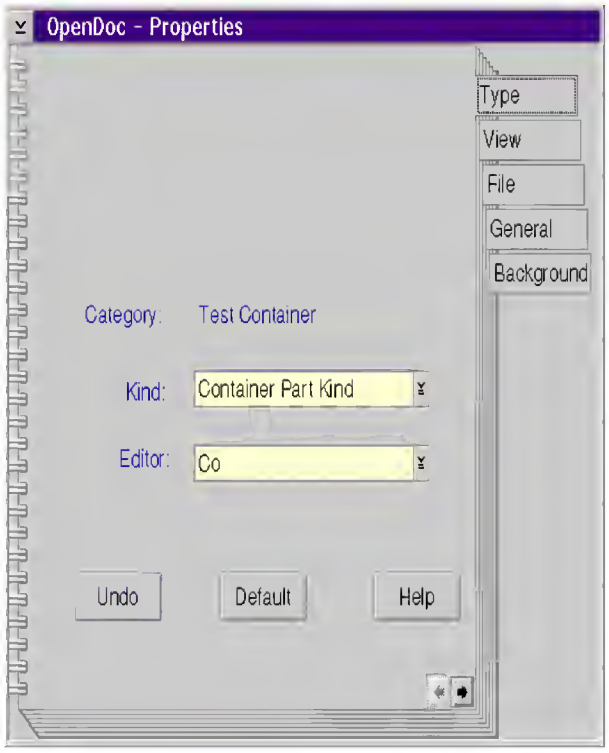
Developers can choose to add pages to the Properties notebook. For example, pages that specify the background color of a window or whether a grid is on or off.

Multiple Frames and Properties

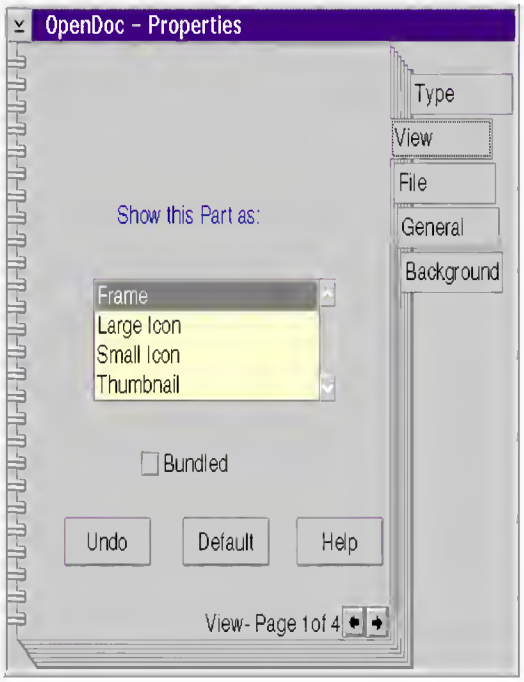
If a part is represented by more than one frame, each frame should have its own Properties notebook. The properties of the object should be divided into object properties (properties that affect the data) and view properties (properties that affect how the data is displayed). Changing object properties will be reflected in all frames of the part. Changing view properties will change just the frame from which the properties choice was chosen. If multiple frames exist for a single part, and Properties notebooks are opened on each frame, each frame will show its Properties notebook. Any change to the object portion of the notebook will immediately be shown in all of the notebooks for that part. It is up to the parts to decide what properties are view properties. In the basic set, **Show as** is the only view property.

Viewing and Changing Part Properties

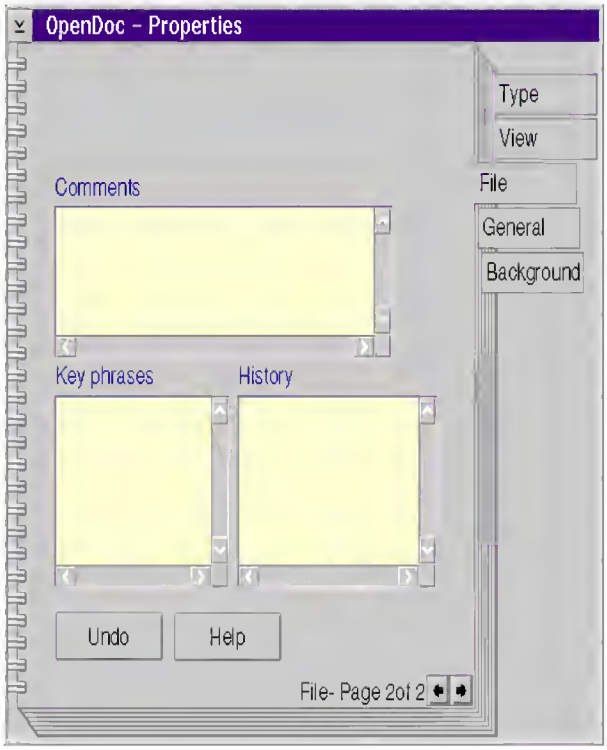
To view and change properties for a part, the user selects **Properties** from a menu associated with the part. A Properties notebook appears, in which the user can see the part's current properties, and make changes to the user-modifiable properties. Properties that change in OS/2 are applied immediately. However, in OpenDoc, changes are applied when the notebook is closed.



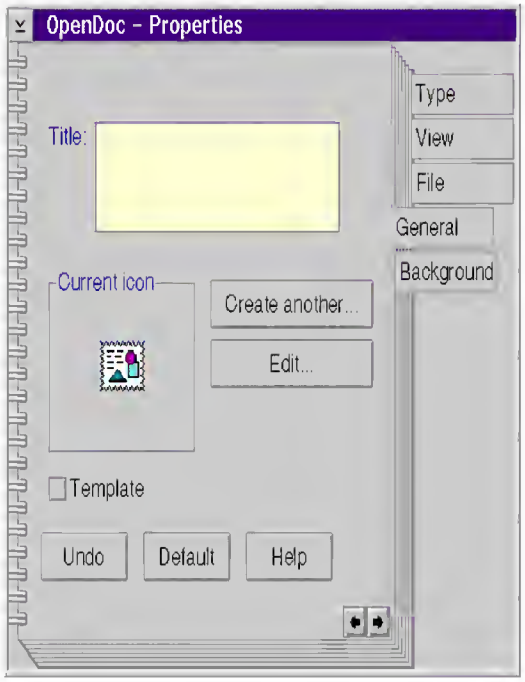
Part Properties Notebook - Type page



Part Properties Notebook -View page 1 - Show as



Part Properties Notebook - Comments page



Part Properties Notebook - General page

Changing Icons

A part's icon is changed in the same manner as a WorkPlace Shell icon. Developers can either use the **Create another**, **Edit**, or **Find** push buttons or any icon can be dropped on top of the current icon within the General page of the Properties notebook, and it will change to the dropped icon.

Renaming a Part

The user can rename parts either by modifying the Name attribute in the Properties notebook for that part, or by directly editing the part's icon label.

The second method is an optimization for the standard first method. This is an example of an accelerator to change a single property. Developers can add others that make sense for their data kind. All containing parts need to support icon text placement and manipulation.

Part Properties Notebook

Every part, whether it is a root part or an embedded part, has its own properties notebook. The Properties notebook can be accessed at any time. To change the properties for a root part, the user would choose **Properties** from the **Document** menu. To change the properties for an embedded part, the user would select **Properties** from a pop-up menu, or activate the part and choose **Properties** from the **View** menu.

Part-Editor Properties

Part-editor properties control the behavior of the editor used to edit a particular part. In addition, the part editor can provide choices that affect the part contents as a whole, rather than some specific selected content. For example, a particular editor may support specialized printing

options. OpenDoc refers to these properties as *settings* . The available settings and the mechanism for changing the properties varies according to the editor being used. Editor properties appear as additional pages in the part's Properties notebook. This section describes the general mechanism that should be used, but developers should decide on the set of properties to support.

Properties of the Part Editor

Each part editor is responsible for presenting its properties to the user through the Properties notebook. The editor pages will be added to the Properties notebook of each part associated with this editor. To modify properties, users use the Editor pages of the part's Properties notebook.

Settings of the Part Editor

Note: This function is not available in this release. This is for information only.

Because one editor may support dozens of settings while another might support none, OpenDoc does not specify a required set of settings. However, there are some settings that will be common to a large percentage of editors. These settings are called the *Standard Set* .

The Standard Set includes:

- Print Options

  The Print Options dialog is provided by the editor application in conjunction with the printing software.

- Script

Inclusions List

If a part developer finds it necessary to disallow certain kinds of parts from being embedded in the part being developed, the developer can use an Inclusions list to specify which kinds of parts are allowed to be embedded. This information would be a page in the part's Properties notebook. It is not recommended that part developers set up Inclusions lists since the purpose of OpenDoc is to allow different types of data to co-exist in the same document. The default is everything allowed.

Editor Properties

Note: The following information is for planning purposes only, it is not available in this release of OpenDoc.

Each editor is represented by an Editor object. Editor properties can be changed by opening the Editor object and changing one of the editor properties. The Editor object can also be used to uninstall the editor from the system. If a user attempts to delete this object, a dialog box will be displayed asking if the editor should be uninstalled from the system.

Editor Preference Object

There will be one object that contains all of the installed editors and their preferences. The name of this object is Editor Preferences. If a user opens this object, a Properties notebook will be shown with a list of kinds and their associated editors.

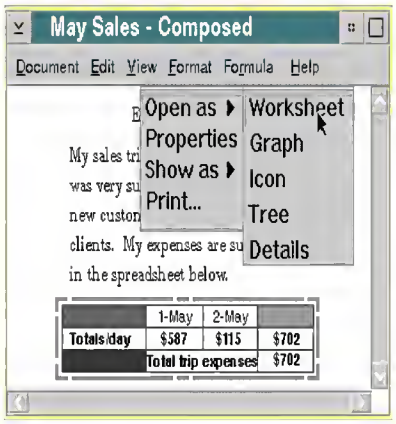
Interactions

Users interact with OpenDoc parts in several ways. This chapter describes those interactions and selection techniques.

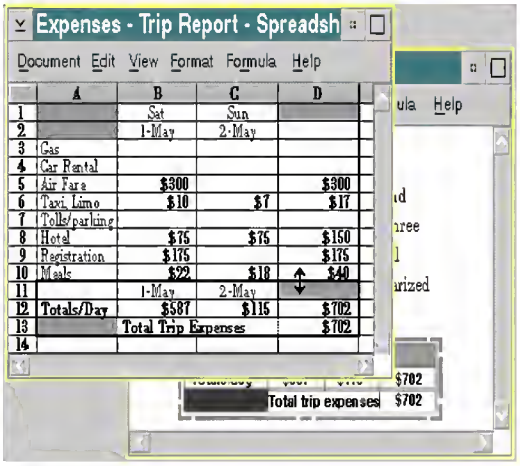
Displaying Parts

From the WorkPlace Shell Desktop, objects must be opened into windows in order to be edited. But in OpenDoc, it is not necessary to open a part into its own window to edit it. A part displayed as frame can be edited in place. However, any part can also be opened into a separate window. When a part is embedded, choosing **Open as** from an associated menu opens the part into a window of its own. Double-clicking on an icon, frame, or the background of a part can be used as an accelerator for the **Open as** {default view} choice. Selecting **Close** in the Window menu (or double-clicking on the Window menu graphic) closes the window. When the window is opened, its size and location are the same as the last time it was closed; if this is the first time the window is opened, default values are used.

If a part is opened into a window, and then **Cut**, the window is closed and is not re-opened at paste time. However, if such a frame is moved (by drag-and-drop) within the document, the window remains visible. A window showing a selected frame's contents. shows a part frame opened into a window. This provides two simultaneous views of the part's contents.



Opening a selected part by choosing Open as



A window showing a selected frame's contents

Any part should support opening into a window of its own. It is up to the developer to supply the **Open as** view choices.

Changing Part Representations

A unique characteristic of OpenDoc is that parts can be represented as either icons or frames. This section describes how to change the

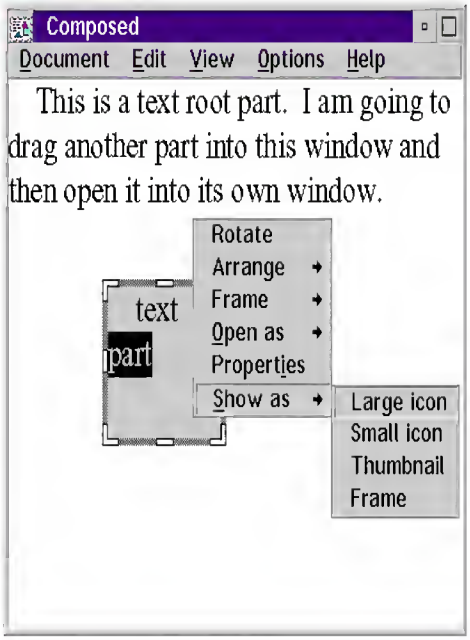
representation of a part from a frame to an icon and an icon to a frame. Developers should provide the necessary information to support Icons view, Tree view, or Details view from a container.

Show As Frame

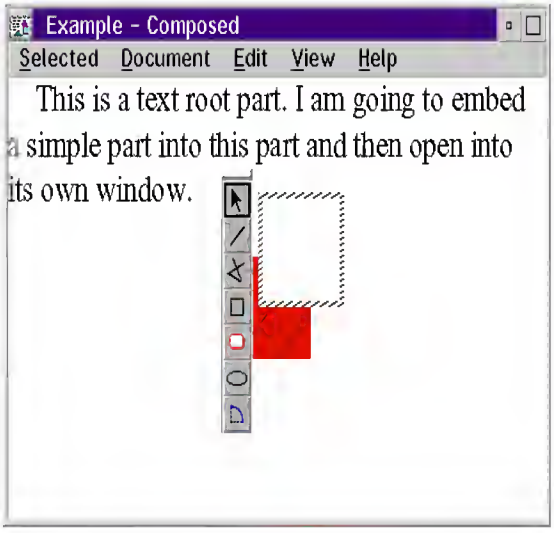
To show a part as a frame, the user can choose **Show as -> Frame** from an associated menu, or change the **Show as** attribute to **Frame** in the part's Properties notebook.

Show As Icon

To show a part as an icon, the user can choose **Show as -> Icon** from an associated menu or change the **Show as** attribute to Icon in the part's Properties notebook.



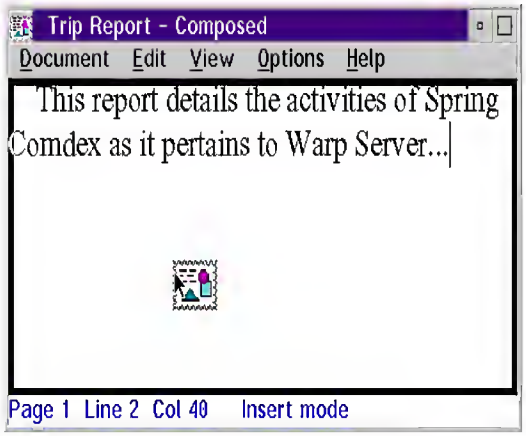
Show as chosen from a pop-up menu



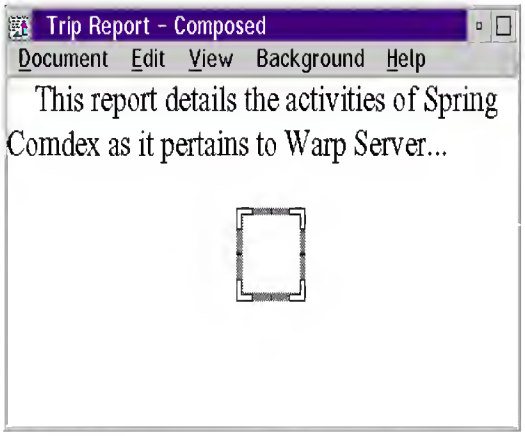
The selected part appears as a frame, replacing the icon

Changing a part's representation causes one representation to be replaced by another. After a representation change, the location of the part generally remains the same, although some containers can change it, such as a graphics part that could align it to a grid. The containing part determines whether or not the layout of its contents (for example, text) is adjusted to accommodate the change. If the representation change results in a frame, as much of the content as can fit is displayed inside the frame. The frame's size is the same as the last time it was displayed; if this is the first time, a default size is used. The user can adjust the size by selecting the frame and using the sizing borders.

Changing the representation of an embedded part has no effect on any views it might have opened in other windows. For example, when changing the representation of a frame to an icon, the part remains open and an icon appears where the frame used to be with the appropriate in-use emphasis. See A part's icon being dragged into a text document. and The part is represented as a frame inside the document..



A part's icon being dragged into a text document



The part is represented as a frame inside the document

Athough any part can be represented as a frame, at a given time a part might not have that ability because its current editor or viewer does not support displaying the part's content. Such parts do not have a frame representation, and the **Show as -> Frame** operation is not available for them. Whenever such a part is asked to display itself as a frame, for example, when it is dragged into a document window, it remains an icon. An example might be a sound part that can be played but not displayed when its icon is double-clicked. However, switching editors for the sound part might allow it to be displayed.

Hierarchy of Preferred Representation for Parts

Every container view has a preferred representation for the parts it contains. Preferred representation determines whether contained parts are represented as icons, small icons, thumbnails, or frames in the containing part. For example, when a part icon is dropped into an open document, the part displays itself as a frame. Likewise, when a part frame is dropped onto the Desktop its representation changes to an icon. When a part is placed into a container, it will take the container's preferred representation for its contained parts. However, the user may change the contained parts representation through its Properties notebook. The developer determines the initial setting for a container's preferred representation.

When a part is placed into a container whose preferred representation is different from the current representation of the part, the part's representation should change to match its container's preferences. If the part does not support the container's preferred representation it should use its current representation. If the current representation cannot be used, the part should be represented by an icon. In any case, the

setting for **Show as** should accurately reflect the part's current representation. Where possible, containers should allow users to override this setting on a part-by-part basis. For example, if a container part is set up in icon view, and a user selects one of the embedded parts and changes its representation to frame, the container should honor this change.

Activating Parts

In OpenDoc, selection is explicitly set by the user, while activation is implicit wherever possible. When a user makes a selection, the selection's container automatically becomes active.

The active part controls the preferred editor, the menus, and the container properties. See {Active Part} Menus for information about active part menus.

Selecting Parts

Selecting parts in an OpenDoc container is analogous to selecting desktop icons in the OS/2 WorkPlace Shell.

In the OS/2 WorkPlace Shell, a user can manipulate objects without first selecting them. By default, mouse button 1 (MB1) is mapped to selection, and mouse button 2 (MB2) is mapped to manipulation. However, users can change these defaults. Parts should not look for raw mouse events, rather for semantic level messages. This concept is followed in OpenDoc.

The following recommendations should be followed for selection for parts themselves. Selection within parts should follow the appropriate guidelines in the *Object-Oriented Interface Design - IBM Common User Access Guidelines* .

Table 1. OpenDoc Part Selection

GOAL	USER ACTION
Select one part	Ctrl+click MB1 or Shift+click MB1
Promote selection from part to container(**)	Alt+click MB1 on part; repeat Alt+click MB1 to select container
Select one or more contiguous parts	Press and hold MB1 outside of the area to be included; stretch marquee box to completely enclose the desired element(s); release MB1
Select multiple parts	Ctrl+click MB1 on each part
Select multiple contiguous parts from an ordered set	Shift+click MB1 on first part; Shift+click MB1 on last part
Add part to existing selection set	Ctrl+click MB1 on unselected part
Remove part from selection set	Ctrl+click MB1 on selected part

As described in Part Icons and Part Frames, a part has two possible types of representations: icon and frame. In addition to the methods described in OpenDoc Part Selection, containers should also support the following representation-specific selection methods.

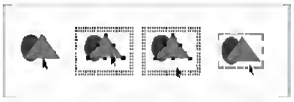
Selecting a Part Icon

To select a part icon the user places the pointer over the icon and clicks MB1. The icon becomes selected and displays Selected emphasis.

Any kind of icon (large icon, small icon, thumbnail) can be selected in any kind of container. An icon can be selected even when its part is opened into a window. The user simply clicks MB1 on the icon.

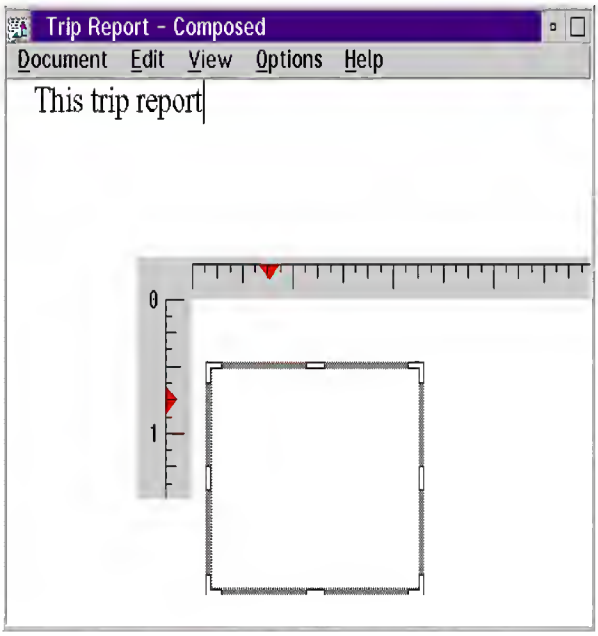
Selecting a Part Frame

Selection in OpenDoc follows an inside-out model; that is, a simple MB1 click selects the lowest-level selectable item under the pointer. In user-editable parts, this simple click will select some content of a part. If a user wants to select an entire part rather than its content, he can click MB1 on the active frame border.



Selecting a frame by clicking on the active border

For most parts when the user clicks MB1 in a part frame, the frame becomes active. However, if the part does not support activation, then it becomes selected and its parent container is activated. Parts that do not support activation are those that have been bundled or those that are provided as simple parts.



Unselected and Selected Frames

Note that unselected, inactive frames such as the ruler can have invisible borders.

When a frame is selected, its container becomes active and displays active emphasis.(**) A frame can be selected even when its underlying part is opened into a window.

Selection can include only contents of the active part. All members of the selected set must be at the same level of embedding within the active part. Selection cannot extend outside of the active part.

In addition to these selection techniques, developers can provide other frame selection methods that are appropriate to the types of parts. For example, a graphics part might provide a lasso tool, a text part might allow selecting with the cursor keys, and a folder part might allow selecting by typing the first letter of embedded parts' names.

Selecting a Part's Contents

Users do not have to explicitly activate a part before selecting its contents. Regardless of what part is currently active or what the current selection is, they can make a new selection by clicking in the window or frame of any part.

OpenDoc follows an inside-out model in determining what to select. When the user presses the mouse button inside a part, the lowest level element of content under the pointer is selected. For example, if the user clicks on some text in a label frame inside a chart frame that is inside a graphic frame, the user gets an insertion point in the text in the label and the label frame becomes activated. If the click is not inside a frame but is at the top level of a window, the top-level part handles the click.

Today, different applications use different selection schemes, depending on the type of data. In OpenDoc, developers define the selection methods for a part's contents. However, OpenDoc proposes a set of uniform selection guidelines for parts in order to improve the consistency of selecting.

Selecting Multiple Parts

OpenDoc Part Selection demonstrates the techniques used in OpenDoc for selecting multiple parts.

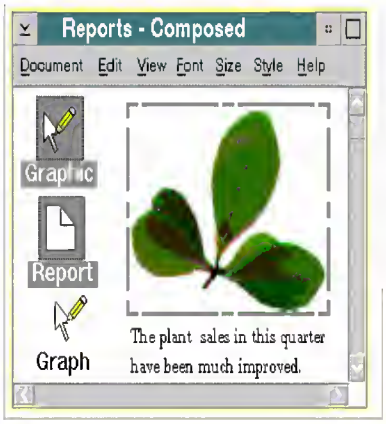
The following rules apply to multiple selection:

- The selected icons and frames must all be in the same active containing part.
- All concurrently selected parts must be siblings (at the same level of embedding).
- Selection cannot exist inside two parts simultaneously.
- There is only one selection set per window.
- When a window becomes inactive, the selection should remain visible, using 50% white in the selection emphasis, to allow dragging of contents and parts.

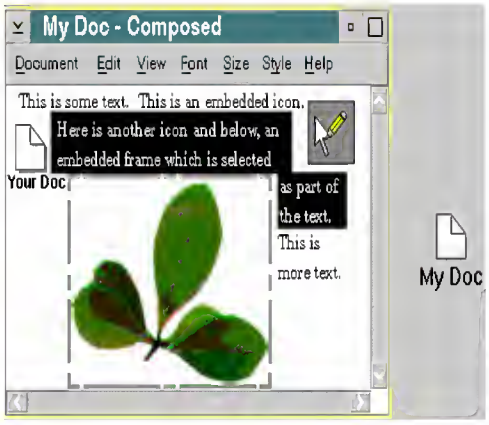
Clicking MB1 anywhere in a part's contents (not in a scroll bar, menu, or palette) with no modifier keys (**Shift**, **Alt**, or **Ctrl**), resets the selection.

All multiple selections are considered to be intrinsic content selections and follow the editing rules for them (see Cut, Copy, Paste, and Create).

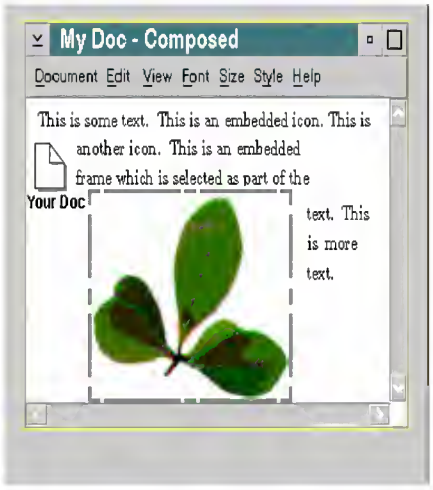
Icons and frames can be freely mixed within a container since they are both representations of parts. Therefore, a multiple selection might include a mixture of icons, frames, and intrinsic contents. Consider the following examples:



A mixture of icons and frames selected in a container

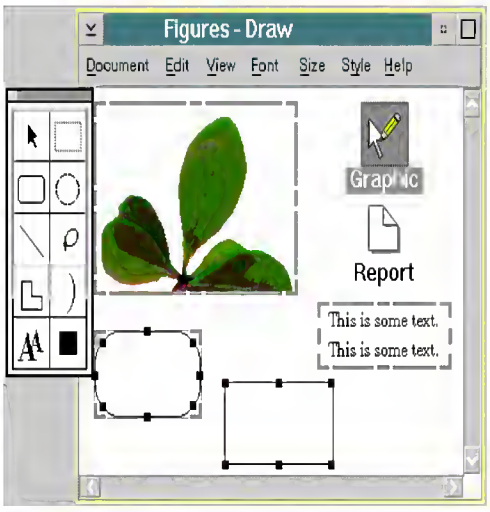


An extended text selection including an icon and a frame



The same frame selected by itself in the same text container

In a graphics container the mechanism for showing selections might be very different:



Example of a multiple selection

Selecting All the Contents of a Part

To select all the contents of a part the user should do the following:

1. Activate a part.
2. Select **Select All** from the **Edit** or pop-up menu.

All the contents of the active part become selected. If the active part is the top-level part in a window, all the contents of the window become selected.

Note that selecting all the contents of a part is not the same as selecting the part itself. If users want to insert the contents of a part into another part, they should select all the contents. If they want to embed a part into another part, they should drag or paste the part as a whole.

Selecting Hot Parts

Certain hot parts, such as buttons and sounds, execute code or perform some other action when they are clicked on with the selection button

(MB1). This means they cannot be selected with an unaugmented click of MB1. However, MB2 can bring up a pop-up menu to perform actions on them, or a user can:

1. Press and hold down the **Alt** key.
2. Click MB1 on the hot part.

Or:

1. Press and hold down the **Ctrl** key.
2. Click MB1 on the hot part.

Or:

1. Press and hold down the mouse button while the pointer is outside of the hot part.
2. Move the pointer over the hot part, thereby defining a marquee rectangle.

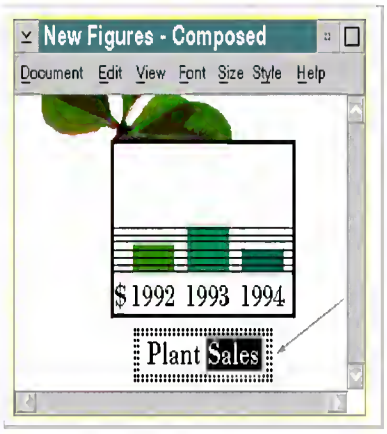
In text, the user can place the mouse pointer just to the left of the hot part and drag to the right side of it, thereby selecting the part without clicking on it. Note that intervening items might become selected. The user might have to toggle the unwanted items out of the selection.

Activating a Frame

OpenDoc considers the container of the contents being edited to be the active part. There are well defined rules for making a part active or inactive and for providing visual feedback so that the user knows which part is active.

To activate a part the user should click MB1 in an unbundled frame. See Selecting Parts about the **Alt**+MB1 selection process.

As with selecting part contents, OpenDoc follows an inside-out model in determining which frame to activate: when the user clicks the mouse button, OpenDoc activates the lowest-level frame that contains the pointer location. For example, if the user selects some text in a label frame inside a chart frame inside a graphic frame, the label frame becomes active, as shown in the following figure:

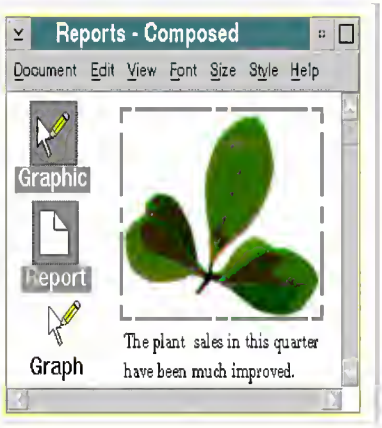


Activating the label frame by selecting text

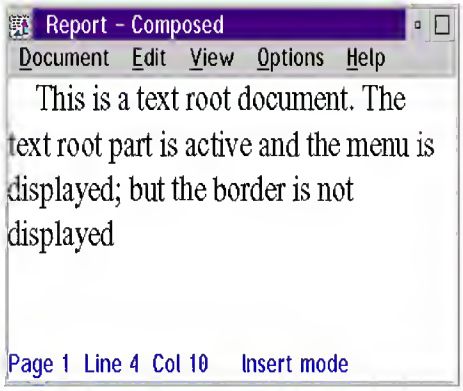
Generally, a frame becomes active and its border appears as soon as the selection mouse button is pressed in it. However, there are exceptions, such as when the pointer is over a bundled frame, in which case the frame is selected.

When a part is activated, its menus, palettes and other user interface features are displayed. In addition, if it is represented as a frame, the frame's border is displayed, as shown in An embedded graphics part is selected. However, if a part is the top-level part in a window its frame border is not displayed. The part's active state is indicated by its active window border (see A text top-level part is active and its menu is displayed).

When selection is begun in an embedded part that is not active, the active border moves to the part that contains the selection. See Selecting Parts for more information.

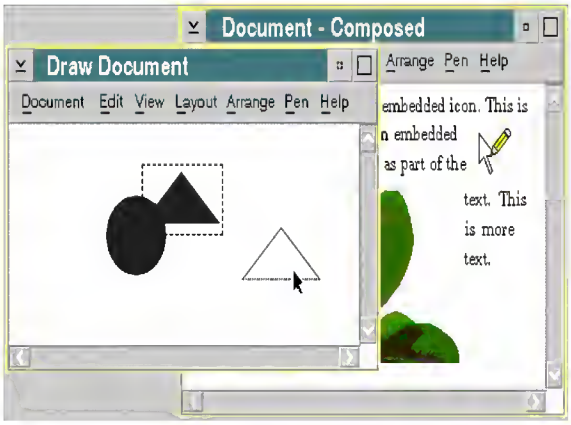


An embedded graphics part is selected and its border, menu, and palette are displayed

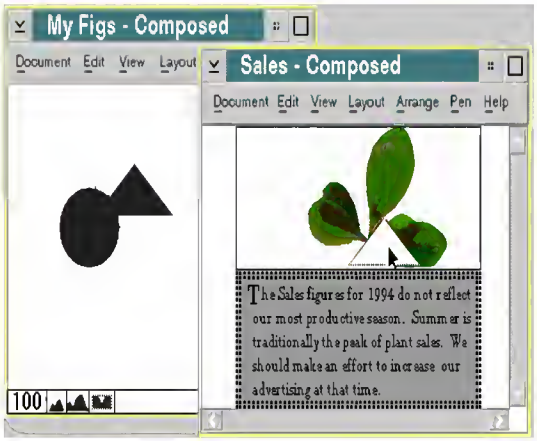


A text top-level part is active and its menu is displayed but its border is not displayed

During the drag operation, as soon as the pointer enters a frame, the receiving part displays target emphasis. This border serves as target feedback for the drag operation. The appearance of the insertion cursor (if any) is controlled by the container, as is the exact placement of the dropped part.

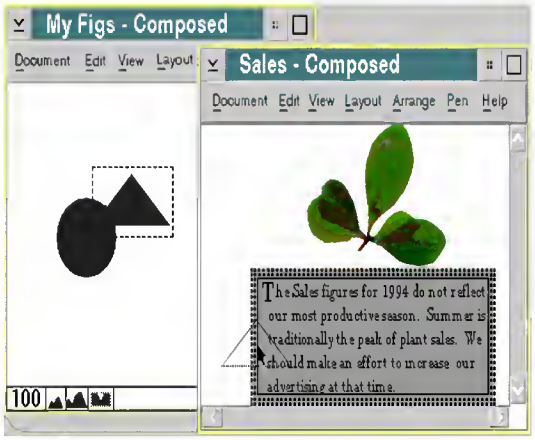


Dragging a triangle

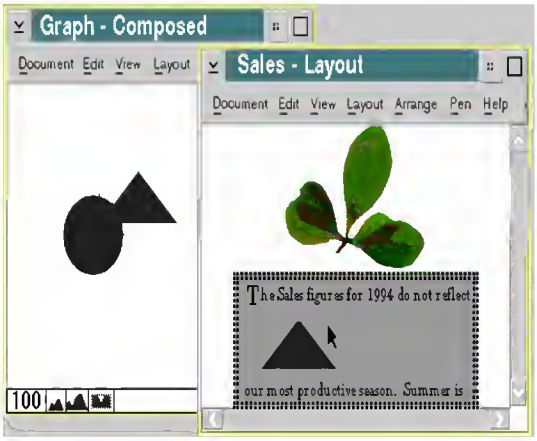


The pointer and triangle enter second window

The part activation is unchanged and the target emphasis is around the leaf graphic part.



Target emphasis moves to the text frame



The triangle is embedded within text part

The text part controls making room for the drawing part.

When the mouse button is released, the dragged item is inserted in the active frame, and target emphasis disappears. Note that drag and drop should not change selection or activation.

Some frames have valid target regions that are smaller than the entire frame. In this case the user sees the target emphasis only within the valid target regions. If the user attempts to drop an item in an invalid location, the item is rejected, returns to its original location, and remains in whatever selection state it was in prior to the drag.

Keyboard Navigation

Part developers should implement keyboard navigation to support activation for the following key combinations:

- **Alt+UpArrow** and **Alt+DownArrow** moves the cursor (and activation) up and down in the hierarchy of part containment.
    - **Alt+UpArrow** moves activation up (activates the currently active part's container) and places the cursor where it last resided in the newly active container. Repeated **Alt+UpArrows** continue to promote activation until the top-level part in the window is active. When the top-level part is active, **Alt+UpArrow** should have no effect.
    - **Alt+DownArrow** (when the cursor is over a part that supports internal navigation) should move activation down one level and place the cursor in its last (or default) position. **Alt+DownArrow**, when the cursor is not over a part supporting internal navigation, should have no effect.

Deactivating a Frame

To deactivate a frame the user can click MB1 outside the frame border within the client area of its container window.

The active frame becomes inactive. Its border disappears, and it no longer receives keyboard events. The lowest level frame in which the pointer is now located becomes active. Clicking in the menu bar, scroll bar, tool bar, or other user-interface element does not deactivate a frame; only clicks in the contents of some other part do so.

Surfacing Windows

Whenever the user selects something in a window, either in the top-level part of the window or in an embedded part, the window containing the selection becomes the active window, and is brought to the top of the stack of windows on the screen so that it is completely visible.

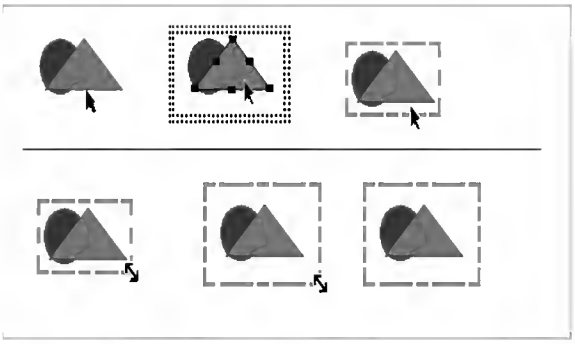
Editing Frames

This section expands on operations that apply to frames. Note that a part might have more than one frame, as described in Part Frames.

Changing the Frame Size

To change the size of a frame the user should do the following:

1. Select a frame.
2. Position the pointer over a resize handle on the border.
3. Press and hold down either mouse button.
4. Move the pointer to a new location.
5. Release the mouse button.



Changing a frame's size by dragging a resize handle

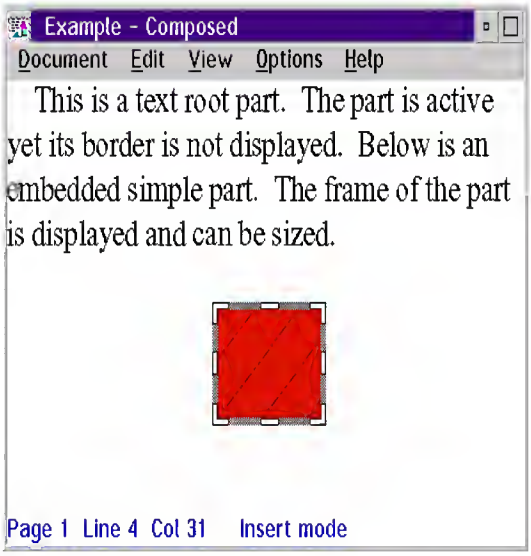
In addition to this direct-manipulation technique, a frame's size can also be changed programatically, for example, by a script. See Scripting for more information about scripting. The containing part determines how many resize handles appear when a frame is selected. Some containers might display fewer resize handles than others, perhaps none if the container does not allow the frame's size to be changed. For example, in a graphics part, an embedded part might have eight handles, while in a text part it might have four.

In addition to controlling how many handles appear, the containing part also controls how much space a frame occupies. When a frame attempts to change size, it negotiates with its container about the new size. The container can grant the requested size, reduce it, or refuse altogether, depending on its current contents and other constraints such as gridding. The container also determines whether to adjust the layout of its other contents (for example, reflowing text), when a frame's size changes.

Pointer Feedback for Changing Frame Shape

The resize handles provide pointer feedback to help the user understand the location and behavior of the border. As the user moves the pointer over any of the handles on a selected frame, the pointer shape changes to the OS/2 double-pointed resizing arrow. This shape indicates that the frame's size may be stretched or reduced from that point. The new frame shape appears as an outline until the resize has been completed, and the mouse button has been released.

Some containers can require that embedded frames be rectangular; their resize handles resize only their rectangular areas. Others can allow embedded frames to be non-rectangular. These containers are responsible for putting an appropriate number of resize handles on a frame's border and interpreting drag actions on them.



A simple part frame with a rectangular border

The Visible Region of a Frame

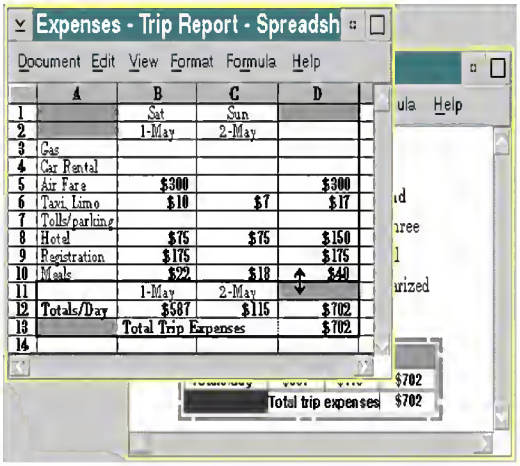
The visible region of a frame is a graphic indication (indicated by an outline) of the portion of a part's contents that is displayed in a frame. The

visible region is indicated by an outline within the window opened on an embedded part.

To change the visible region of a frame the user should do the following:

1. Open a frame into a window.
2. If the frame outline is not already visible, Choose **Show Frame Outline** from the **View** menu in the new window; the frame's outline appears indicating the region visible in the container's window.
3. Drag the frame outline to include the desired portion of the part window's contents.

The newly-included portion of the part will appear inside the part's frame in the document window.



Setting a frame's visible region

The frame on the right has been opened into the window on the left.

When the pointer is over the outline, the cursor shape changes to a moving arrow. The user presses MB2 on it and drags it to move the area of inclusion in the visible region. The frame's contents change dynamically as the frame outline moves around. The user can also choose **Hide Frame Outline** from the **View** menu to disable the display of the outline.

In the prevous figure, the document on the right contains text and a spreadsheet frame. The user has opened the spreadsheet frame into a window. The user has selected the **Show Frame Outline** choice in the **View** menu. The part's visible region indicator appeared. This is the full-intensity rectangle in the lower cells of the window. The user has placed the pointer over this border; the cursor shape changed into a resizing arrow. The user then drags the frame outline within the window to change the visible area. It is constrained to stay within the window. Since the frame is visible on the screen, the composed document dynamically updates to show whatever the frame outline currently includes.

A visible region outline has the same size and shape as the associated frame. Its size and shape cannot be changed directly. The only way to change them is by changing the frame itself (with resize handles) within the container's window. This is because the containing part determines its embedded part's frame size.

Cut, Copy, Paste, and Create

The **Cut**, **Copy**, **Paste**, and **Create** choices are familiar to users of GUIs. These choices are the traditional mechanisms used to copy and move content around in the system. Refer to the *Object-Oriented Interface Design - IBM Common User Access Guidelines* for additional information about these functions. In OpenDoc, these choices continue to operate similarly, but with some additional functionality, to handle embedded parts.

The additional functionality consists of:

- Operating on parts as well as intrinsic content. The parts can be represented as frames, large icons, small icons, or thumbnails.
- Making embed-versus-incorporate decisions.
- Wrappering intrinsic content inside a part in certain circumstances.

The user will usually want the system to make the decisions automatically when performing these functions. However, the user can override these decisions by choosing a new menu choice, **Paste As...**. This choice is described later in this section.

If the user copies or moves a part, the copy or move contains all the data currently in the part, including any edits that have not been saved.

The clipboard choice, **Create** (similar to the **New** choice in many of today's applications) has been added allowing users to create new parts and place them on the clipboard.

Create Choice

The **Create** choice is used to place a new part on the clipboard of the same kind as the part selected. This differs from **Copy** in that a new part is created that does not include the content of the source part.

To place new part on a clipboard the user can select a part of the desired kind and:

1. Choose **Create** from an associated menu.
2. Choose a target.
3. Choose **Paste** from the **Edit** or pop-up menu of the target.

Or:

1. Use drag-and-drop equivalents (see Drag-and-Drop Data Transfer Techniques).

Note: If multiple parts are selected, a new part is created for each selected part.

Copy Choice

Copy is used to copy intrinsic data or make a new part of the original part's kind that carries the original part's contents.

To copy parts, the user chooses **Copy** from a menu associated with the original part.

1. Select a part or part contents.
2. Choose **Copy** from the **Edit** or pop-up menu.
3. Select a target.
4. Choose **Paste** from the **Edit** or pop-up menu.

Or:

1. Use drag-and-drop equivalents (see Drag-and-Drop Data Transfer Techniques).

Cut Choice

The **Cut** choice is used to remove selections and place them in the Clipboard so that they can later be pasted.

To move selections to the clipboard:

1. Choose **Cut** from a menu associated with the desired part or intrinsic data.
2. Select a target.
3. Choose **Paste** from the **Edit** or pop-up menu.

Or:

1. Use drag-and-drop equivalents. Drag the selection to the clipboard (see Drag-and-Drop Data Transfer Techniques).

When the **Cut** choice is selected, the cut item disappears from its old location. When the **Paste** choice is selected, the cut item appears at the new location. If the source and target are the same part, the **Cut** and **Paste** operation simply repositions it. If the source and target are different parts, the cut items change containers.

Copy or Move Within a Part

Copying or moving content within a part amounts to editing the contents of a single part and therefore is largely the province of the part developer. Each part controls the editing of its intrinsic contents.

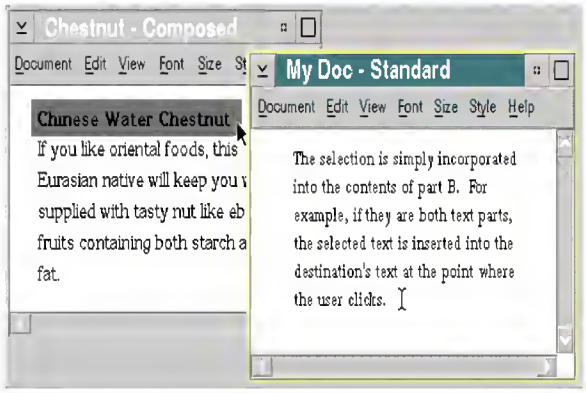
Copy or Move Between Parts

The following section describes the default behaviors of **Copy** or **Move** between parts.

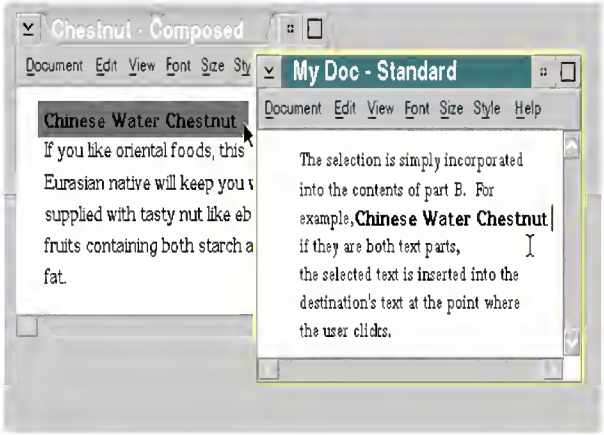
Every part has its own kind of intrinsic contents. For example, text parts contain text characters, graphic parts contain graphic elements. The following cases illustrate the default results of copying or moving content between parts. See Incorporate versus Embed Default Decision Summary for a summary of these defaults.

- Case 1

  The source and target parts are the same kind of part. The selection is simply incorporated into the contents of the target part. For example, if both parts are text parts, the selected text is incorporated into the target's text at the insertion point, and becomes intrinsic content of the target part. The next two figures illustrate this behavior.



Selected text copied between documents of the same kind

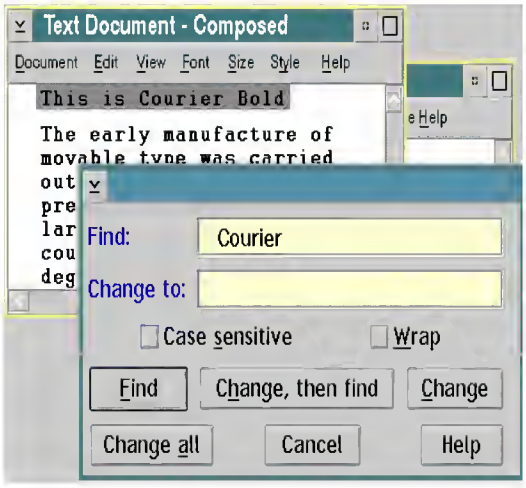


A copy of the text is incorporated into the target

In the previous example, the text was copied to My Doc and inserted into the contents of that document. Note that the source is still selected. The newly inserted text is not selected because if it were, the user might accidentally type over the new paste. If the target had been any other kind of part, for example graphics, the pasted text would have remained selected as detailed in Case 3.

- Case 2

  The source and target parts are different kinds, but the target part can handle the category of the selection. For example, if the source and target are text parts that store text in different formats, the selected text is converted to the target's format and then inserted into the target text. A common situation is copying text from a rich text editor, such as SurfWriter Text, to the simple text in a dialog box; the user does not want a part to be created and embedded within the dialog box text.



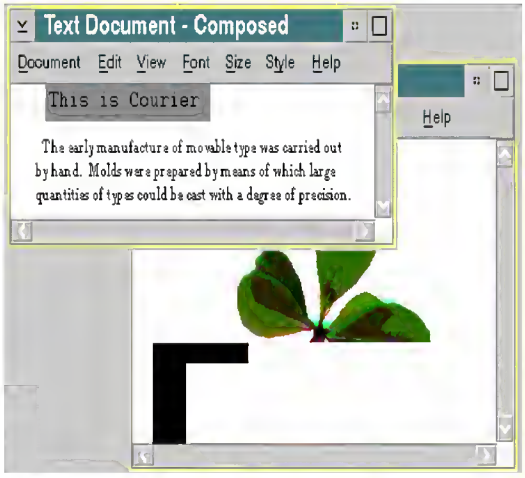
Text converted to plain text

The plain text is inserted into the dialog box text part - its style properties are stripped out and it reflows itself.

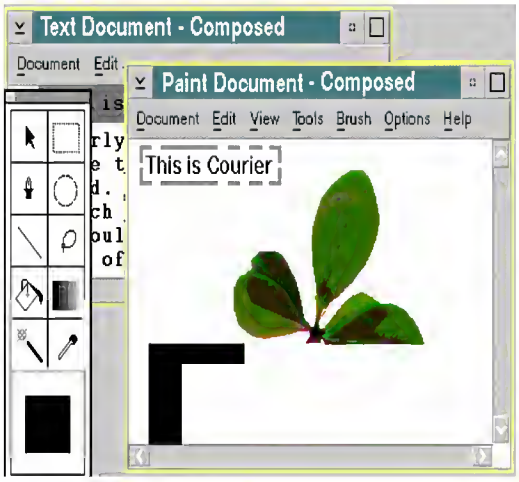
- Case 3

  The source and target parts are different kinds, and the target part cannot handle the category of the selection. OpenDoc creates a new part of the same kind as the source part, inserts the selection in it, and embeds the new part in the target part. For example, some text pasted into a painting would not be converted into a collection of bits. Instead, the text would be encapsulated inside a text part. Then the text part would be embedded into the target.

This protocol supports the goal of preserving, wherever possible, all relationships among moved elements, including spatial relationships and links. For example, if several circles and rectangles are selected and moved, they have the same appearance at the target regardless of which of the above cases applies. However, preserving spatial relationships is not always possible; for example, text content is usually reflowed.



Text in Text Document-Composed is selected and copied



A copy of the text is encapsulated in a new text part

The new text part is the same kind as the source. The new part is embedded in the graphics document, and the new part is then selected.

In the previous example, because the text was embedded into a part other than text, it cannot be deleted by over-typing. Therefore it remains selected after insertion to facilitate adjustment of its position, size, etc.

Note: Pressing the **Alt** key during a drag operation forces the dragged content to be embedded as a separate part in the containing part. For example, if a text part's content is being dragged into another text part, pressing the **Alt** key during the drag operation will force the dragged content to be embedded as a text part rather than be incorporated.

Copying or Moving a Single Part

Suppose the user copies a single whole part to another part, either by using the Clipboard or by dragging it. The entire selected part is copied and embedded as a unit into the target part. The copied part's structure as a part is preserved. If the target part is of the same kind as the source container, any container-assigned properties are changed to conform to the new container; these properties can vary from container to container.

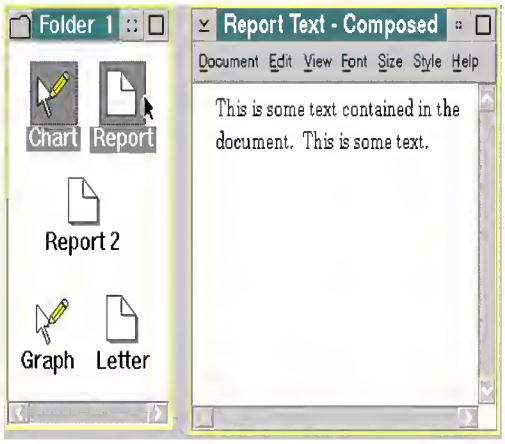
The part takes on the preferred representation specified by the target. Thus, if the target prefers to represent its contents as frames, the copied part appears as a frame. Likewise, if the target prefers to represent its contents as icons, the copied part appears as an icon. See Hierarchy of Preferred Representation for Parts for more information.

The intent of making single part selections a special case is to give users the ability to embed any part in any other part, regardless of the kinds of parts. This is handy in Cases 1 and 2 on pages Copy or Move Between Parts and Copy or Move Between Parts, in which the copied part is compatible with the target part. For example, if the user wants to embed a SurfWriter text part into a FooWrite text part, the user can select the Surfwriter part as a whole and copy or move it into the FooWrite part. The Surfwriter part is embedded as an independent part in the FooWrite part, preserving its nature as a Surfwriter part. Alternatively, the user can merge the text of a Surfwriter part into a FooWrite part by selecting the contents of the Surfwriter part (for example, with **Select all**) and copying or moving them into the FooWrite part. The Surfwriter text is converted to FooWrite text and inserted into it.

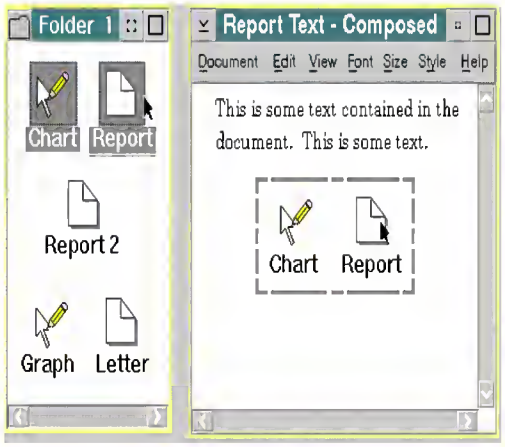
Copying or Moving Multiple Parts

When the user selects more than one part, with or without any of the containing part's intrinsic content, this selection is treated as an intrinsic-content selection.

If the target is the same document as the source, then the parts are simply repositioned in it. If the target is on a removable drive or a LAN drive, then the parts are copied. The target may adjust the specified location based on constraints such as gridding in a graphics part (or where text exists in a text part). If the user presses MB2 anywhere within a selection, the complete selection is dragged. Even if the pointer is over an embedded part within the selected scope, OpenDoc assumes the user wants to want to drag all the selected objects and not just the part under the pointer. For instance, the selection is encapsulated in a wrapper part of the same kind as the containing source part, and the new part is then embedded in the target part. This preserves the spatial and other relationships among the selected parts. See the following figures.



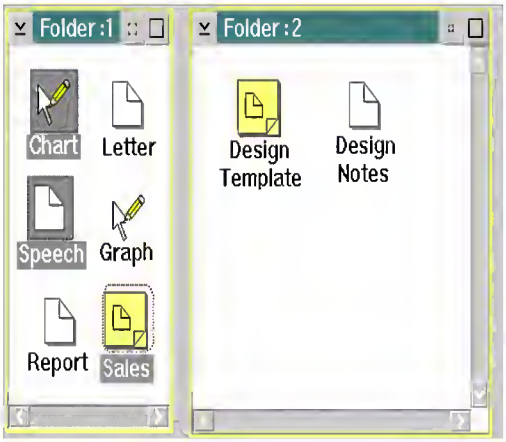
Copying a multiple selection from a folder into a text document



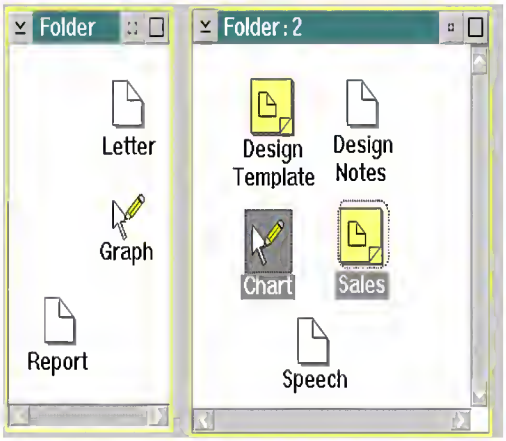
The multiple selection is encapsulated in a new folder part

The new folder part is inserted into the text.

- In Cases 1 and 2 on pages Copy or Move Between Parts and Copy or Move Between Parts, the wrapper part is not necessary and the selected parts are embedded directly into the target part. For example, if several part icons are copied from one folder window to another, they appear as separate icons in the target folder, because the source and target are the same kind of part. See the following figures.



Moving a multiple selection from one folder to another



Source and target parts of the same type

Because the target is the same type of part as the source, the selection need not be encapsulated in a new folder part.

Thus, there is a difference between copying a single part and copying multiple parts. The single part is always embedded as the default action (although the user can override this action with the **Paste As** dialog option) while multiple parts and intrinsic content can be embedded or incorporated, depending on the source and target kinds.

These behaviors of moving between parts are summarized in the following table. The first column lists various selections one could make in a part, the second column compares the target and source parts, and the third lists how the selection is deposited in the target.

Table 2. Incorporate versus Embed Default Decision Summary

SELECTION	TARGET PART	ACTION
intrinsic contents	same kind as source part	incorporate
intrinsic contents	different kind, compatible category	translate and incorporate (not in this release)
intrinsic contents	different kind, incompatible category	encapsulate into same kind of part as source and embed
single part (any kind)	any kind	embed
multiple parts (any kinds)	same kind as source part	incorporate
multiple parts (any kinds)	different kind, compatible category	translate and incorporate (not in this release)
multiple parts (any kinds)	different kind, incompatible category	encapsulate into same kind of part as source and embed

---

Copying or Moving Onto an Icon

If it makes sense for the target part, the user can drop something onto a part icon (a closed part), just as is done with closed folders on the WorkPlace Shell. It is up to part developers to support this functionality. The dragged parts or data are placed inside the part at some default location determined by the part (usually at the end of its contents). If the intended target part cannot contain the source part being dragged onto it, the target part will not allow the drop and the source part will remain where it was before the drag was attempted.

Moving Frames

To move frames the user should:

1. Select the frame.
2. Position the pointer on the background of the frame; then press and hold down MB2.
3. **Move** the pointer to a new location, dragging the frame.
4. Release the mouse button.

Or:

1. Select a frame so that its selection border is visible.
2. Position the pointer over the selection border (but not over a resize handle).
3. Press and hold down MB1 or MB2.
4. Move the pointer to a new location.
5. Release the mouse button.

Or:

1. Position the pointer anywhere inside a bundled frame.
2. Press and hold down MB2.
3. Move the pointer to a new location, dragging the frame.
4. Release the mouse button.

Or:

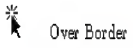
1. Select a frame.
2. Select **Cut** from the **Edit** or pop-up menu.
3. Click in a target part.
4. Select **Paste** from the **Edit** or pop-up menu.

Dragging will probably be the preferred method, since it involves only the mouse.

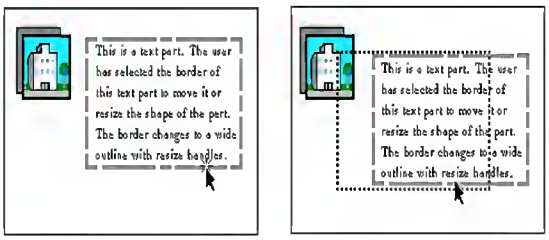
Feedback for Moving Frames

To help users understand how they can manipulate frames, OpenDoc containers should provide visual feedback. For example, when the user starts to drag the frame, the frame should change to a dotted outline. This reduces the possibility that the frame would obscure any underlying content as the user repositions the frame. This moving frame feedback is identical for any frame regardless of its current selection or activation state.

The dragged outline will retain its original size and shape while it is being dragged. When a frame is dropped it will take on the preferred content representation of its current container. For example, if a frame is dropped onto the Desktop, it will turn into a icon (the Desktop's preferred representation for its contents).



Pointer over border



Frame border drag representation

The frame's border is represented as a dotted line while it is being dragged.

Moving a Template Part

Templates behave differently from other objects. When you drag a template, the default action is **Create** another (a new object is torn off). To move a template by direct manipulation, the user must press **Shift**+drag (which is the explicit move combination). See Template for more information.

To move a template part using menu operations, the user should do the following:

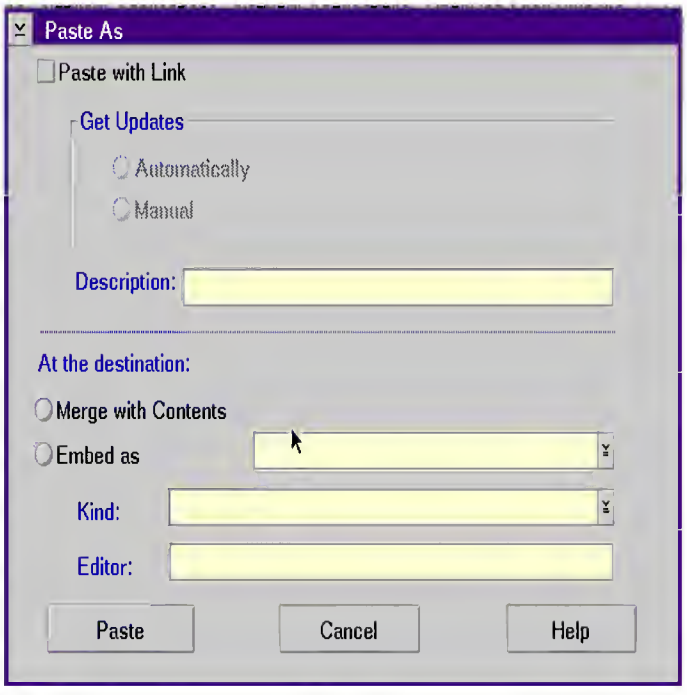
Note: These are the same operations used to move any part by using menus.

1. Select a template.
2. Select **Cut** or **Copy** from the **Edit** or or pop-up menu.
3. Select a target.
4. Select **Paste as** from the **Edit** or pop-up menu.

Using the Paste As Choice

The **Paste as** choice is used for:

- Controlling data embedding during **Move** and **Copy** operations.
- Controlling part representations during **Move** and **Copy** operations; for example, icon or frame.
- Creating links.



The Paste as Dialog Box

The contents of the clipboard are pasted at the specified location according to these options. The meanings of the options follow:

Paste with **Link**

A check box. If the user chooses **Paste with Link**, a link is created between the source and target. See Drafts, Linking, and Scripting for more information about **Paste with Link**.

Get Updates

Contains two radio buttons (**Automatically** and **Manually**) specifying when changes to the source cause the target to update. **Automatically** means the target will be updated as soon as the change is received. **Manually** means only when the user selects the Update Now choice from the Properties notebook of the **Link Target** page (see Selecting Linked Parts or Data for more information about these choices); this inhibits the target from displaying changes to the source when it receives them, effectively disabling the link. The default is **Automatically**.

Description

Allows the user to identify the link with a description of it.

At the target of a link, the user can make the following choices:

Merge with Contents

A radio button paired with **Embed as**. These choices allow the user to control the embedding of data in new parts.

Normally, during **Move** and **Copy** operations, data is automatically embedded in a new part if it is of a different kind than the target part. The **Paste as** dialog gives the user manual control over this embedding. It is also the primary means for creating links.

Merge with Contents interacts with the **Kind** option in the following way.

If a Kind is specified that the target part cannot handle, **Merge with Contents** is marked as unavailable and **Embed as** becomes selected. On the other hand, if the selected Kind can be handled by the target, the user can choose **Merge with Contents** and select a Kind from a drop-down list of kinds the target part can handle. Thus the user can never ask a target part to accept a kind of data that it does not support.

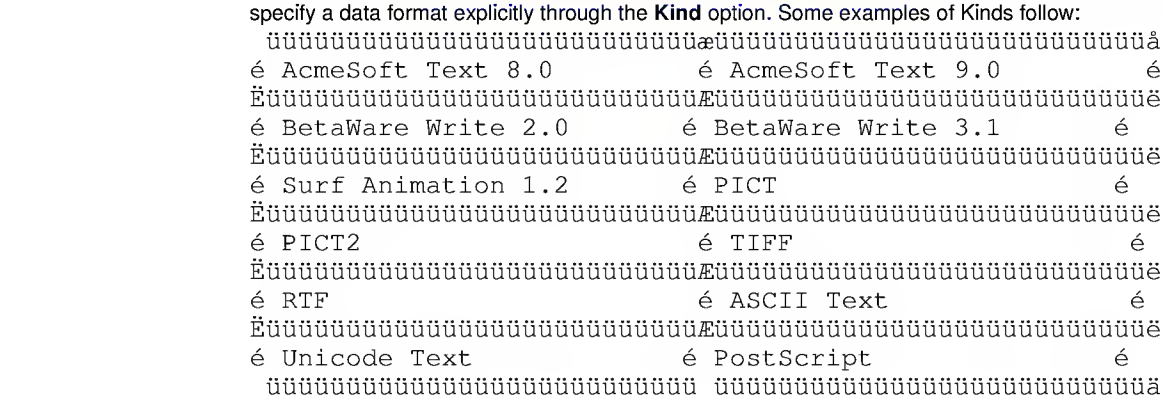
Embed as (Frame, Large Icon, Small Icon, Thumbnail)

Ordinarily if the source and target kinds are compatible the source part would be merged with the target. However, the user might want to maintain the pasted selection as a separate part so that it can be manipulated separately from its container. In this case, the user would choose **Embed as**. For example, the user might wish to embed a text sidebar in a text page. The **Embed as** option forces the source to be embedded in the target as a separate part. When **Embed as** is selected, the entire list of Kinds appear.

If the target document is restricted to allow only certain kinds of parts to be embedded in it, the **Embed as** option might not be valid. In this case, it is marked as unavailable. The drop-down list allows the user to specify the representation of the part that will be embedded, that is, a frame, a large icon, a small icon, or a thumbnail. Also note that links can be embedded or merged with the data as well.

Kind

The format in which the pasted data will appear. The available kinds are presented as a drop-down list of choices. These choices allow the user to control the format of pasted data. Normally if data has to be converted to a different format during a **Move**, **Copy**, or **Link** operation, the active part will choose will choose a default format according to the rules described in Editors and Viewers and Part Kinds and Categories. But the user can override this default and specify a data format explicitly through the **Kind** option. Some examples of Kinds follow:



Editor

A read-only field displaying the part editor name.

Paste

A push button to select the paste operation.

Drag-and-Drop Data Transfer Techniques

Users can copy and move items within OpenDoc without using menus by dragging the items with the mouse pointer. Dragging is performed with the manipulation button (the default in OS/2 is MB2).

OpenDoc follows the drag defaults specified by the *Object-Oriented Interface Design-IBM Common User Access Guidelines* . Unmodified dragging in certain contexts implies **Move**, while unmodified dragging in other contexts implies **Copy**. The terms Copy-drag and Move-drag are used to clarify that they are the drag-and-drop versions of the **Copy** and **Move** operations.

When an OpenDoc document from the Desktop is Copy-dragged into another document, it will create a new part with the same name and content as the one on the Desktop. Likewise, Copy-dragging a part to the Desktop will create a new document with the part's name. If the document name already exists in the same container, the current rules for name clashes of WorkPlace Shell objects will be followed. The name will be appended with **:** and a number starting with 1.

When a user of OpenDoc drags an item within a document, the item is moved. When the user drags an item across document boundaries, the item is also moved unless the drag is between fixed media and removable media (for example, a diskette)(**), in which case it is copied.

Occasionally, the user might wish to override the system defaults and force a **Copy** or **Move** operation, so mechanisms for doing so are provided. The **Ctrl** key is used to force a **Copy** operation, while the **Shift** key is used to force a **Move** operation.

The following user actions illustrate the default and forced operations.

To accept default behavior:

1. Position the mouse pointer on the source part.
2. Press and hold down MB2.
3. Move the pointer to a new location.
4. Release the mouse button.

To force a copy:

1. Position the pointer on the source part.
2. Press and hold down the **Ctrl** key.
3. Press and hold down MB2.
4. Move the pointer, dragging the source, to a new location.
5. Release the mouse button.
6. Release the **Ctrl** key.

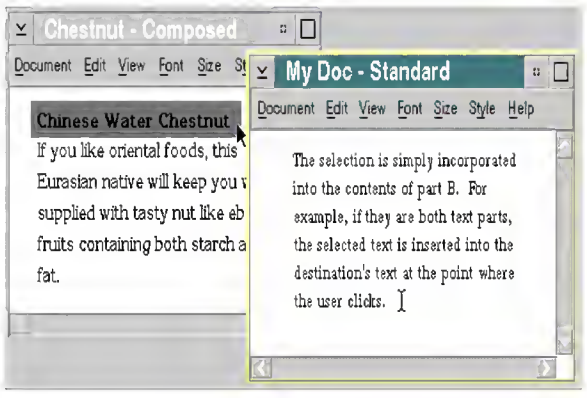
   The system makes a copy of the dragged item and places the copy at the pointer location. The **Ctrl** key may be pressed anytime. The state of the **Ctrl** key at the end of a drag-before the mouse button is released is significant. A Drag-move can be turned into a Drag-copy at any time prior to releasing the mouse button by pressing the **Ctrl** key.

To force a move:

1. Position the pointer on the source part.
2. Press and hold down the **Shift** key.
3. Press and hold down MB2.
4. Move the pointer to a new location.
5. Release the mouse button.
6. Release the **Shift** key.

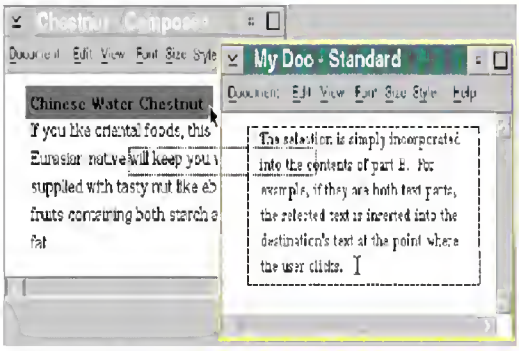
Regardless of whether the user is performing a drag-move or drag-copy, an outline of the source is dragged to the new location; it follows the pointer as the mouse moves. When the mouse button is released, the item is placed (dropped) at the pointer location or at the nearest default drop point. Whatever part the pointer is over receives a drop event notifying it that something has been dropped on it. Each receiving part should follow the Embed versus Incorporate rules discussed in Cut, Copy, Paste, and Create.

Using drag-and-drop can be easier than using menus, but it requires that parts provide additional user feedback. The next three figures show dragging some content between documents of the same kind.



Selected text to be copied between documents of the same kind

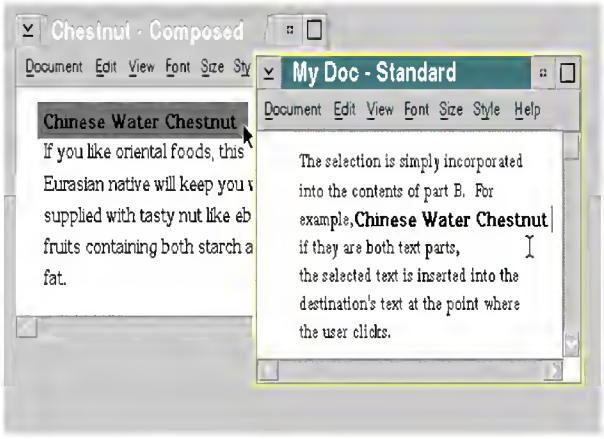
The first three words of the document *Chestnut* have been selected and are about to be copy-dragged. The next figure shows the drag feedback *hat is the outline of the selection, which shows that this selection is being dragged into the document *My Doc* .



Text content from Chestnut is being dragged to My Doc

Note that drag-and-drop does not allow content to be dropped into a part if that content is not of one of the types on the Inclusions List for that part.

To indicate the target of a drop, normal WorkPlace Shell target emphasis will be used. In the case of the root-part (or document), the target emphasis appears just inside the window border (the same happens when you drag over an open WorkPlace Shell folder), Because this window shows target emphasis, the user knows which part the content is being dragged into. If the part cannot accept the drop, the cursor should change to the OS/2 *do not drop* indicator.



A copy of the text is incorporated into the target

A copy of the text is incorporated into the target. shows that a copy of the three words dragged from *Chestnut* have now been dropped into *My Doc* . The target part now behaves exactly as though **Copy** and **Paste** operations had been used instead of drag-and-drop.

Accelerators

An accelerator is a mnemonic and a key or combination of keys that a user can press to perform an action that is available from a menu. Developers should support predefined mnemonics and accelerator key assignments for each predefined choice in a menu. Developers should also provide a unique shortcut key assignment for each frequently-used choice in a menu. See the *Object-Oriented Interface Design-IBM Common User Access Guidelines* for more information about mnemonics and accelerator keys.

Management of Contained Part Windows

When a contained part is dragged to another location, the result is a copy or move of the underlying part and of its representation (small icon, large icon, thumbnail, or frame) in the new location. Any windows currently open on the target container part will update to show the new content.

When a drag is a regular move within a document, the window of the moved part remains open. However, if the result of the drag is a move outside of a document, any windows that were open, close when the icon is dropped onto its new location, and the information that was in the window is preserved in its current state in the new location.

Creating New Parts

There are several ways to make a new part. These methods can be broken down into two categories, explicit and implicit creation. Parts are explicitly created by tearing off a new object from a template, copying a single part somewhere, or using the **Create** menu choice. Parts are implicitly created as a side-effect of pasting some content from another part. For example, copying content in one part and pasting it into another part of a different kind creates a new embedded part. These are described in detail in Cut, Copy, Paste, and Create and in Template.

Deleting a Part

Users can delete parts by using one of the following methods:

1. Drag a part (icon or frame) to the Shredder icon.

   The part is removed and deleted.

Or:

1. Select a part (icon or frame).
2. Choose **Delete selection** from the **Edit** or pop-up menu. The selected part is deleted. The clipboard is not involved.

Or:

1. Select a part (icon or frame).
2. Press the **Del** key. The selected part is deleted. The clipboard is not involved.

Users can cause parts to be deleted as a side effect of other user actions. Users should be warned before these implicit, or accidental, deletions take effect. The implicit sequences are:

1. Select a part frame.
2. Choose **Cut** from the **Edit** or pop-up menu.
3. Overwrite the clipboard contents by copying or cutting again.

   The selected part is removed from its container and placed on the clipboard. If the user does not Paste before selecting another **Cut** or **Copy**, the previous **Cut** selection is deleted.

Or:

1. **Move** a part (icon or frame) into another part's window, where the target part is in a different document (and the document does not have Perfect Save).
2. Close the target document without saving it.

   This has the effect of deleting the moved part, although the user may not be aware of it. A message is displayed asking if the user wants to save the changes. If the answer is no, then the moved part is deleted.

Or:

1. Select a part (icon or frame) in a text container.
2. Type a character. The selected part is deleted and replaced by the typed character if the container accepts typing as data input. This operation should be reversible.

Parts that auto-save or perfect-save such as folders do not need to be explictly saved when their windows are closed, since closing them does not delete anything. In fact, such parts do not even have a Save choice. Only windows for parts that do not auto-save can cause content to be lost; which includes most document windows.

The user can delete multiple parts at once by making a multiple selection before doing one of the above actions.

Undo

Undo {*action*} is designed to allow recovery from errors that the user recognizes soon after they are made. The **Undo** choice is displayed as:

```
Undo {action that can be undone}
```

Part developers should supply wording that describes all undoable actions. The words are appended to the **Undo** choice to indicate to the user what will happen when they choose **Undo**. For example:

```
Undo typing
```

In most of today's applications, only one level of **Undo** is usually supported. Single-level **Undo** allows the user to correct only the most recent undoable action taken. For example, if the user pastes to a target that was not the intended target and then types a word, the typing is the only action that can be undone. This limitation renders the **Undo** ineffective.

For this reason, OpenDoc supports multiple levels of **Undo** and **Redo**. This change allows users to recover from more errors than is possible today. For example, if the user invokes **Undo** three times in succession, the last three undoable actions are undone in reverse order. There is no inherent limit in the number of times that **Undo** can be invoked in succession. However some user actions clear out the **Undo** history, resetting it to zero. A typical example is a user selecting **Save** from the **Document** menu; **Undo** cannot unsave it, or undo any actions taken prior to the **Save** action. Developers determine what actions reset the **Undo** history for each part.

To use **Undo** the user chooses **Undo** from the **Edit** menu.

The following scenarios illustrate how **Undo** behaves when the user uses drag-and-drop:

Scenario 1:

1. User selects text in document 1 and Copy-drags it to open document 2. (Copy-drag refers to the force **Copy** operations described in Drag-and-Drop Data Transfer Techniques).
2. User selects the **Undo** choice. The inserted text in document 2 is removed. The text in document 1 is left undisturbed because it was merely copied.

Scenario 2:

1. User selects text in document 1 and drags it (a default move) to an open window on document 2.
2. User selects the **Undo** choice. The inserted text in document 2 is removed and the text reappears in document 1.

Note: Closing any window resets the system's **Undo** history.

The effects of the last undoable user action are reversed, restoring all parts to their states before that action. Not all user actions are reversible. For example, scrolling, making selections, and opening and closing windows are not reversable, because they do not change the contents of a part. In general, developers decide what the undoable actions are.

Some actions in OpenDoc might affect two or more parts. For example, move-dragging something from one part to another changes both parts.

Note: If both parts are within the same document, then **Undo** is supported across the parts. However, for this release of OpenDoc, **Undo** is not supported across documents. Thus errors that are made between two different documents or between a document and the WorkPlace Shell are not undoable.

Redo

Redo is a choice on the **Edit** menu that reverses the last **Undo** action that has not been redone. It is available only if the last reversible user action was to invoke **Undo** or **Redo**. **Redo** restores all parts to their states before the **Undo** action was invoked. The **Redo** choice is displayed as:

```
Redo {action that can be undone}
```

Part developers should supply wording that describes all undoable actions. The words are appended to the **Redo** choice to indicate to the user what will happen when they choose **Redo**. For example:

```
Redo typing
```

Note: The developer should use the same words to describe the **Redo** action as were used in the **Undo** action.

The following table illustrates an extended **Undo** - **Redo** sequence:

ÜÜÜå

Table 3. Extended Undo - Redo	
USER TRIES TO DO	SYSTEM ACTION
action A	does A's effects
action B	does B's effects
action C	does C's effects
Undo	reverses C's effects
Redo	does C's effects
Undo	reverses C's effects
Undo	reverses B's effects
Redo	does B's effects
action D	does D's effects
Redo	invalid, can not be chosen
Undo	reverses D's effects
Undo	reverses B's effects
Save	writes the changes to disk
Undo	invalid, can not be chosen

OpenDoc supports multiple levels of **Redo**. Every time the user selects **Redo**, the next **Undo** action, if any, is reversed. Repeated **Redo** works only if the user selected **Undo** multiple times in succession. For example, suppose the user selects **Undo** three times in a row. The user can then select **Redo** up to three times in a row to restore those actions, but he cannot select **Redo** four times since there were not that many **Undo** actions.

If the user performs a new action other than an **Undo** or **Redo**, then **Redo** can no longer be performed because the **Redo** history is cleared.

Sources for Parts

This section describes how users access new and existing parts. In addition, Installing and Removing Editors discusses the ways in which editors are installed onto a user's system.

Two ways in which users can create new parts were previously described:

- Copy a part

  The user copies an existing part. A new part is created of the same type as the original, containing a copy of all its contents.
- Create a part

  The user selects **Create...** from the object or pop-up menu. A new part is created of the same type as the original, but without the original's contents.

Another source for parts is:

- Templates

  The simplest and most frequently used method for creating a part is to drag off a part from a template. The new part is of the same type as the template and may contain a specified subset of the template's contents.

Template

Conceptually, a *template* is modeled after a pad of paper in the physical world. When you want a new sheet of paper, you tear it off the pad. Templates can also contain data (or other parts). Users can also use templates to create preconfigured objects. For example, a business-letter part newly dragged off a template may already have the sender's address filled in, as well as the current date. It could also have the margins and other text-formatting options set as appropriate for such a document. Developers can use templates to provide useful items such as business letters with preset areas for the header, address, greeting and body of the letter.

In OpenDoc, a template is a part that has the Template property set. This setting changes the drag behavior so that an unaugmented (default) drag from a template causes the creation of a new part.

The new part is not a template because its Template property is off. The new part is an ordinary part of the same type as the template. The new part contains a copy of all of the contents of the template. (**)

Creating a Template

To convert a part into a template:

1. Choose **Properties** from a pulldown or pop-up menu.
2. Choose the **General** tab.
3. Click in the check box labeled Template.

Creating a New Part from a Template

To create an object from a template the user should do the following:

1. Open the OpenDoc Templates folder.
2. Position the pointer on a template icon.
3. Press and hold MB2.
4. Drag the pointer (with the newly created part icon attached) to a new target.
5. Release MB2.

For example, the user drags a new object from a template in a folder window to a document window. This would cause the new object to appear as a frame. The template part remains an icon in the folder. It is not changed. See The copy appears as a frame..

Moving a Template Icon

To move a template the user should do the following:

1. Position the pointer on a template icon.
2. Press and hold the **Shift** key.
3. Press and hold MB2.
4. Drag the template icon to the new target.
5. Release MB2 and then release the **Shift** key.

Note: This is the same as a forced move for any object.

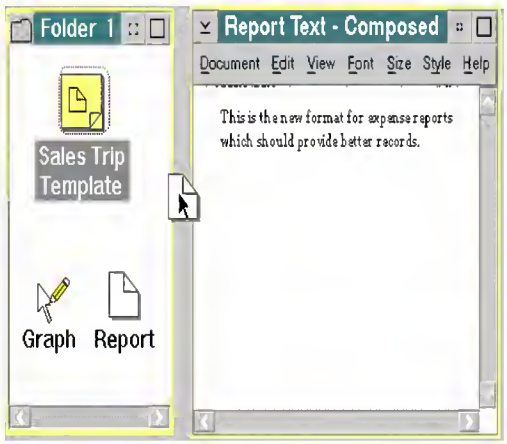
Copying a Template

To copy a template the user should do the following:

1. Point at a template icon.
2. Press and hold the **Ctrl** key.
3. Press and hold MB2.
4. Drag the template icon to the new target.
5. Release MB2 and then release the **Ctrl** key.

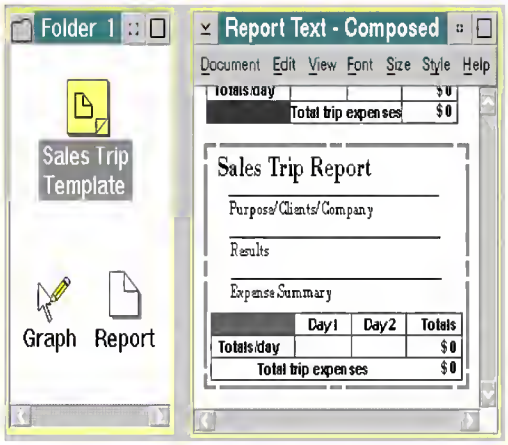
Note: This is the same as a forced copy for any object.

The new copy is exactly like the original; it is also a template.



The icon is dragged into an open window

A new part is placed in the Trip Report window.



The copy appears as a frame

Installing and Removing Editors

This section describes what happens when part's editors and viewers are installed or removed from an OpenDoc system.

Installing an Editor

Users will install new editors for one of two reasons.

First, a user might have acquired a new kind of part for which no editor exists currently on his system. In this situation, it is safe to assume that the user wishes to use the new editor for its kind of part. Thus, the system wide editor preferences are set accordingly. In this case, installation

of the editor should be transparent to the user.

The other reason a user might install a new editor is to update an older version of an editor already in the system or have access to an additional editor. In this situation it is not safe to make the updated editor the preferred editor for this kind, because doing so would cause any existing parts on the system to now be edited using this new editor.

When subsequent editors for the same kind are installed (another text editor), the user is asked if they want the new editor to be installed and if this editor should become the preferred one.

When the user installs a newer version of an editor it should be installed without disturbing any existing versions in the system. Because the part's editor preferences still refer to the original editors, parts in existing documents continue to use their original editors, until the user explicitly deletes the editors(**).

If users wish to substitute editors on a part, they may modify the part's editor preference. System wide preferences may be changed in the Editor Preferences object as described in Editor Properties. In addition, when a part is opened, but its preferred editor has been removed, the user will be asked to specify another editor preference.

OS/2 is currently working on a design to install applications and system function through drag-and-drop. This method will provide developers with a set of tools to create a draggable package to install objects.

The following defaults are recommended for installation:

- All part-editor files should be installed in a directory off of the OS2\OpenDoc directory.
- Templates should be placed in the OpenDoc Templates folder. If a Part Editor provides several part templates (more than 3) then a folder should be created in the OpenDoc Templates folder in order to hold the parts. Users should be able to subsequenly move the templates wherever they prefer to keep them.

Editors and Categories

When an editor is installed OpenDoc checks the category information and determines whether an editor of that category has already been installed on the system. If not, the new editor becomes the preferred editor for all parts of that category, For example, if an editor for the 2D Graphics category of parts was never installed, installing a Draw editor results in setting that editor as the preferred editor for the 2D Graphics category.

Note: If the user has only a viewer installed for a category and then installs an editor, the editor becomes the preferred one for that category.

Replacing Part Editors

The following are situations when a user might want to replace a part editor:

- A user no longer wishes to use a viewer or editor for a particular part type. For example, rather than using AcmeWrite Text for text, the user now wishes to use BetaWare Write. See Choosing an Editor in Editor Properties.
- The user might prefer to use a different editor for handling a specific part. The particular viewer or editor associated with a part is controlled at the part level. The user can select a part and specify a different editor by modifying the part properties.

Drafts, Linking, and Scripting

This section describes the advantages of the drafts and draft-history, linking, and scripting facilities in OpenDoc.

Drafts

A *draft* is a particular version of a document. A series of drafts can serve as a change history for a document, recording the revisions that it

goes through and thereby creating an editing history of the document.

Users can create a new draft at any time to take a snapshot of the current state of the document. Drafts are not created automatically, such as during Save operations. They are created explicitly by users. Drafts are stored efficiently, as differences from the previous draft, so there is little cost for using them.

Drafts are represented as objects in the draft history of the document. Drafts have many of the same behaviors as other objects. A user can read, delete, or copy a draft into a new document. However, users cannot edit drafts in the draft history. The user may use an earlier draft as the basis of a new document by first copying a draft, then editing the copied version of the draft.

Users may examine old drafts at any time. Perhaps they want to see how a document has changed recently. Or they may decide that earlier drafts are incorrect or no longer needed so they want to delete them.

Draft History

The set of drafts associated with a document is called the *draft history* . The information stored in the history includes when changes were made, by whom, and comments about the changes. The draft history does not include the current document because it has not yet been saved as an actual draft. To revert to a previous draft the user must either move or copy the draft somewhere else (to the Desktop for example).

A user can access the draft history of a document by selecting **Draft history** from the **Drafts** cascaded menu (on the Document menu). A new window is opened into a Details view. All the normal folder operations are available except that the **Draft history** view cannot serve as the target of any **Copy**, **Move**, or **Link** operations.
Menus

The following menu items appear on the **Draft history** menu bar:
History

Print draft history...
: Leads to a dialog allowing the user to print a list of the **Draft history** contents.

Edit

Delete
: Deletes the selected draft.

Select all
: Selects all the drafts in the view.

Deselect all
: Deselects all the drafts in the view.

Open selection
: Opens the selected draft(s) into a read-only view.

Selection properties
: Opens a window showing the properties of a selected draft.

Print selection...
: Leads to a dialog allowing the user to print the content of a selected draft.

View

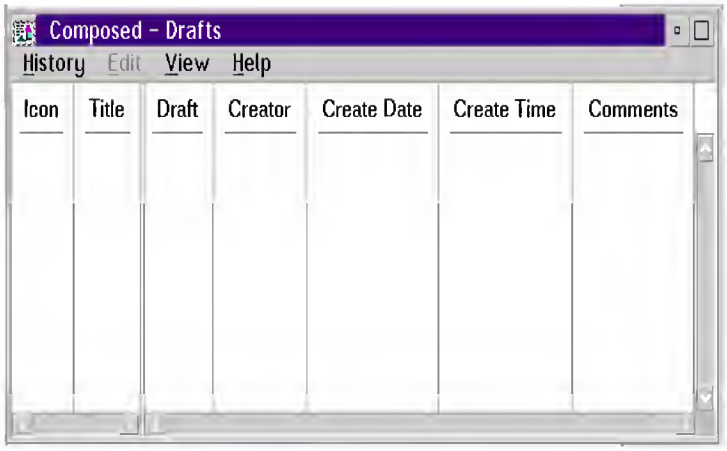
Icons
: Shows icons view in the current window.

Details
: Shows details view in the current window (details view is the default view when **Draft history** is opened).

Help

General Help
: Help on Draft history.

Using Help
: How to use Help Manager.



Drafts View

Drafts are shown as individual document icons, much like files in a folder. Each draft has the following pieces of information associated with it:

Draft name
: The name for the draft. This name will be used as the object name if it is copied to the Desktop.

Creator
: The user who created the draft. For the first draft, this is the user who created the document.

Created
: The date and time the draft was created. This information is separated into two columns.

Comment
: A field the user can set containing explanatory information for other users. The comment is optional and is for the user's information.

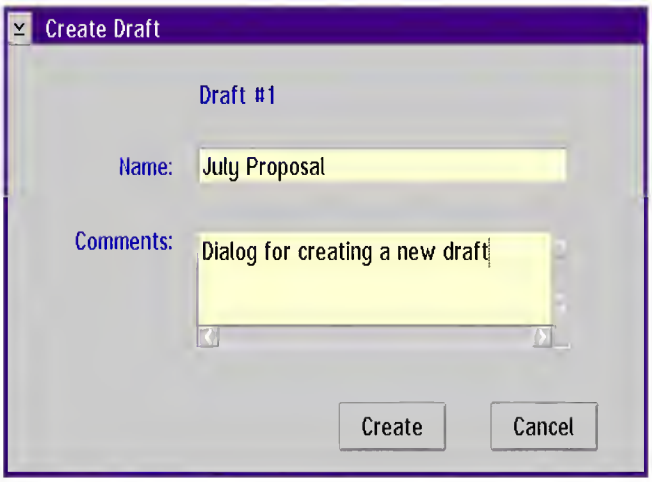
OpenDoc creates most of this information automatically; users supply the comments and the creator name. The window title of the draft history window is of the form {Document name} - Drafts.

Creating a Draft

To create a draft the user should do the following:

1. From the Document window, select **Document => Drafts => Create Draft**.

A dialog appears requesting additional information:



Dialog for creating a new draft

The draft name will default to the (Document name)-Draft (1). The draft number is the next higher unused integer. The last name used will be the default for Creator; however, if the user has a logon name then that should be used as the default creator name. The user can choose to

modify the Creator field, and can also enter a comment to distinguish the new draft from older ones.

When the user clicks on **Create** in the **Create draft** dialog, the current document is saved as a new draft. This action saves the document and the user does not have to explicitly save the document before creating a draft. The information about the new draft is added to the top of the draft history.

The new draft becomes locked; it is no longer the current draft and it can no longer be modified.

If the user clicks the **Cancel** button, the **Create draft** dialog box is dismissed, no new draft is created, and the focus returns to the current document's window.

Editing a Draft

A user can edit only the current document. All other drafts are read-only. Users can select intrinsic contents or parts in the other open drafts but not modify them. Any attempt to modify a draft is rejected with a message. The menus displayed when a draft window is open and active should match those of a viewer displaying a document. For example, functions that cause modifications in the open draft are disabled; however, **Copy** and **Save a Copy** are available.

Users cannot edit the **Creator name**, **Comment**, or contents of drafts.

Deleting a Draft

To delete a draft the user should do the following:

1. Select a draft in the **Draft History** window
2. Choose **Delete** from an **Edit** menu, pop-up menu, or by pressing the **Del** key.

Or:

1. Drag a draft from the draft history window to the shredder.

When one of the these user actions is taken, the standard WorkPlace Shell confirmation is displayed.

If the user proceeds, the draft is deleted from the document's draft history and the draft can no longer be accessed.

If the user deletes the most recent draft, and then creates a new draft, it will have the same number as the previously deleted draft. However, if the user deletes an older draft, its number is not reused.

Note: Deleting a draft may not reclaim all the disk space occupied by the draft, since a document is stored as a series of deltas from previous drafts and these deltas are passed along to the next draft when a draft is deleted.

Linking Documents

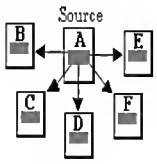
The concept of a linked document is similar to the concept of a database and shadows of that database. Whether a user links a paragraph of text or some type of chart or graphic to another document (or documents/parts), whenever the user updates the original source, the linked target can also be updated.

Linking will be widely used in OpenDoc's compound documents, more so than in today's applications, for two reasons. First, linking is essential to the concept of generic parts; that is, parts that perform services for other parts. For example, by linking parts users can employ a bar-chart part to display data from a table part or a spreadsheet part. Developers do not have to build a charting capability into each of these parts. Linking enables parts to leverage each other. Second, OpenDoc's user interface to linking is simpler than previous linking interfaces.

Link

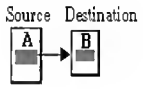
A *link* is a relationship between a source item and a copy of the source item. Unlike an ordinary copy that is a one-time snapshot, a linked copy is updated every time the original item changes. Both part contents and entire parts can be linked. Links can exist within a single part, between two parts in a single document, or between two parts in different documents. For instance, a document might link a picture from another document, or a bar chart might display data that is linked from a spreadsheet.

Links are one way; every link has a single source and a single target:



A Link from A to B

A given source item can be linked many times to different targets, but each connection would be a separate, independent link:



Multiple Links from the Same Source

When a source item changes, all of its links are updated, however, the target of a link cannot be edited.

The source of a link can be any content; for example, a single letter, a sentence fragment, multiple paragraphs, cells in a spreadsheet, a circle in a drawing, a table, or all the contents of a part. The source of a link can also include embedded content. The target can be in the same document as the source (intra-document linking) or in a different document (inter-document linking).

If a source item has multiple links from it, each target can display the data differently. A target may display the data in the same format as the source; for example, a financial report might display a few spreadsheet cells linked in from a larger master spreadsheet. Or a target might transform the data by one of its algorithms. For example, a charting program might graph data from the same cells as a bar chart.

Creating a Link

To create a link using menus the user should do the following:

1. Select a source.
2. Choose **Copy** from the **Edit** or pop-up menu.
3. Click in a target part.
4. Choose **Paste As...** from the **Edit** or a pop-up menu.
5. Check the **Paste with Link** option if not checked or grayed.
6. Click on the **Paste** push button.

Notice that creating a link using the menus method is similar to making a simple copy.

There is an equivalent drag-and-drop method for making a link. The user makes a selection in the source document and drags it to the target with the Shift and Ctrl keys pressed down. When the user releases the mouse button, the link will be created.

To use the drag and drop method, the user should do the following:

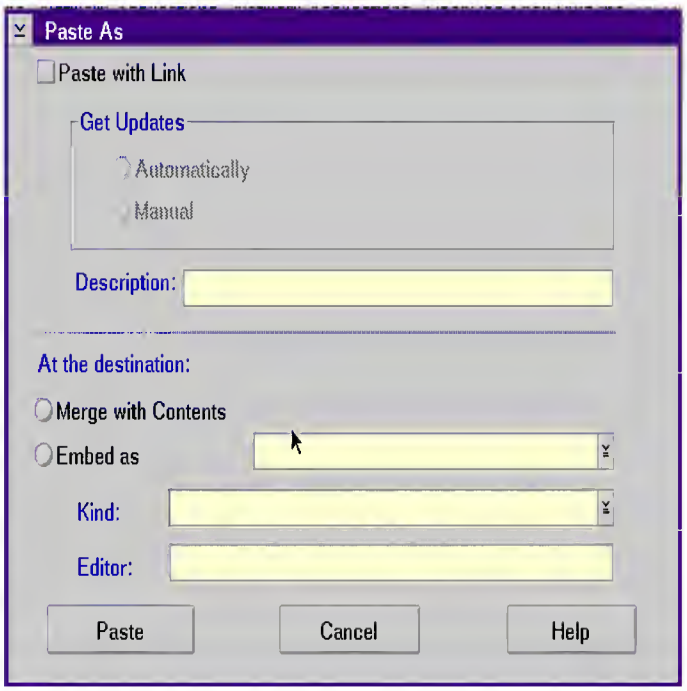
1. Hold down the **Shift** and **Ctrl** keys.
2. Press and hold MB2.
3. Drag the selection into a target part.
4. Release MB2.
5. Release the **Shift** and **Ctrl** keys.

Using Paste As

The **Paste as** menu choice is used for:

1. Creating links.
2. Controlling data embedding during **Move**, **Copy**, or **Link** operations.
3. Controlling how embedded parts appear after **Move**, **Copy**, or **Link** operations, for example, icon or frame.

This choice is described in Cut, Copy, Paste, and Create.



The Paste as Dialog Box

The meanings of the options follow:

Paste with Link

A check box. If the user chooses **Paste with Link**, a link is created between the source and target. See Drafts, Linking, and Scripting for more information about **Paste with Link**.

Get Updates

Contains two radio buttons (**Automatically** and **Manually**) specifying when changes to the source cause the target to update. **Automatically** means the target will be updated as soon as the change is received. **Manually** means the target will be updated only when the user selects the Update Now choice from the Properties notebook of the **Link Target** page (see Selecting Linked Parts or Data for more information about these choices); this choice inhibits the target from displaying changes to the source when it receives them, effectively disabling the link. The default is **Automatically**.

Description

Allows the user to identify the link with a description of it.

At the target of a link, the user can make the following choices:

Merge with Contents

A radio button paired with **Embed as**. These choices allow the user to control the way data is pasted in new parts.

Normally, during **Move** and **Copy** operations, data is automatically embedded in a new part if it is of a different kind than the target part. The **Merge with Contents** choice allows the user to overwrite this default behavior. It is also the primary means for creating links.

Merge with Contents interacts with the **Kind** option in the following way.

If a **Kind** is specified that the target part cannot handle, **Merge with Contents** is marked as unavailable and **Embed as** becomes selected. On the other hand, if the selected Kind can be handled by the target, the user can choose **Merge with Contents** and select a **Kind** from a drop-down list of kinds the target part can handle.

Embed as (Frame, Large Icon, Small Icon, Thumbnail)

Ordinarily if the source and target kinds are compatible the source part would be merged with the target. However, the user might want to maintain the pasted selection as a separate part so that it can be manipulated separately from its container. In this case, the user would choose **Embed as**. For example, the user might wish to embed a text sidebar in a text page. The **Embed as** option forces the source to be embedded in the target as a separate part.

When **Embed as** is selected, the entire list of Kinds appear. Thus the user can never ask a target part to accept a kind of data that it does not support.

If the target document is restricted to allow only certain kinds of parts to be embedded in it, the **Embed as** option might not be valid. In this case, it is marked as unavailable. The drop-down list allows the user to specify the representations of the part that will be embedded, that is, a frame, a large icon, a small icon, or a thumbnail. Also note that links can be

embedded or merged with the data as well.

Kind

The format in which the pasted data will appear. The available kinds are presented as a drop-down list of choices. These choices allow the user to control the format of pasted data. Normally if data has to be converted to a different format during a **Move**, **Copy**, or **Link** operation, the active part will choose will choose a default format according to the rules described in Editors and Viewers and Part Kinds and Categories. But the user can override this default and specify a data format explicitly through the **Kind** option. Some examples of Kinds follow:

```
 üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüå
é AcmeSoft Text 8.0          é AcmeSoft Text 9.0          é
Ëüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Beta Ware Write 2.0        é Beta Ware Write 3.1        é
Ëüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Surf Animation 1.2         é PICT                       é
Ëüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é PICT2                      é TIFF                       é
Ëüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é RTF                        é ASCII Text                 é
Ëüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Unicode Text               é PostScript                 é
 ûüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
```

Editor

A read-only field displaying the part editor name.

Paste

A push button to select the paste operation.

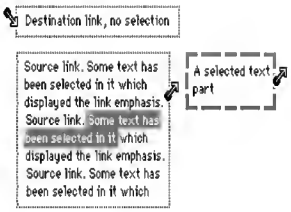
After a link is pasted, its emphasis will remain in place until the user selects the **Hide links** menu choice.

Link emphasis also appears when the user interacts with a linked part; for example, when a user sets an insertion point within a link source or selects some content of a link. In these cases, the link emphasis is shown in addition to the active highlighting of the part's frame. **Link** emphasis that occurs as a side effect of interaction is removed from the part when the user clicks outside of the linked part.

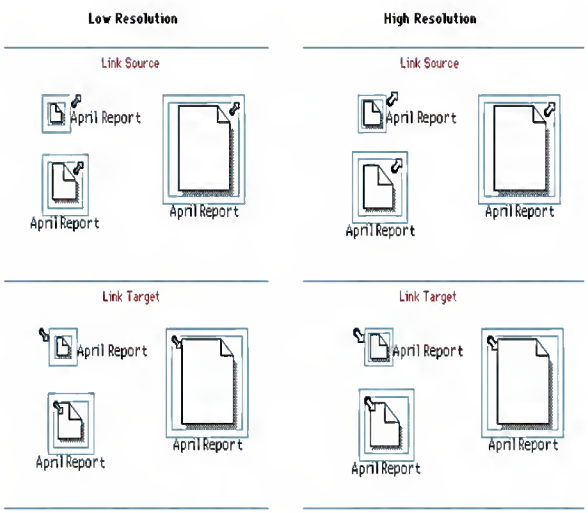
When an inactive, unselected link is highlighted with the **Show links** option, a dotted box is drawn around the frame; otherwise, an active or selected border is used. An outward-pointing arrow is added to the right side of the frame for a source link. For a target link, an inward-pointing arrow is added to the left side of the frame.

All editing choices are available and operate as before for parts that are sources of links; link targets cannot be edited directly. Note that the **Show Links** and **Hide Links** function have no effect on the user's ability to edit links.

The link graphic indicating a link should be added to the active or selected border or to the dotted line that indicates the link boundaries at the bottom of the first quarter of the part's height, starting from the top. This position minimizes the risk of conflict with resize handles. The next few figures show examples of link feedback (also see Appendix A for a detailedview of the link graphic).



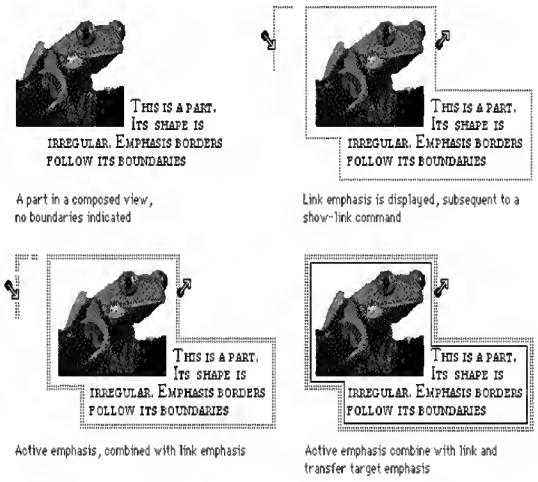
Link Feedback on Frames



Link Feedback on Icons

The boxes around the icons represent the edges of the icons' bit maps and the boundaries of the emphasis region. These boxes do not appear on the user's screen. For more details see pixel-wise views in Appendix B.

This link highlighting overlays (and may obscure) any surrounding content. It does not cause the surrounding content to reformat itself.



Active Irregular Shaped Part with Link Feedback

Instead of repeating the pictures for both target and source links, the source link is shown in its entirety along with a portion of a target link border.

Selected Irregular Shaped Part with Link Feedback

If a window that includes linked parts is closed and later reopened, the link highlighting remains in its previous state. Note that when **Show Links** is set, *all* (**) sources and targets in the active part are highlighted.

Showing the Source of the Link

Link emphasis does not indicate which target is linked to which source. OpenDoc allows target links to be tracked back to the source link, but not vice versa.

From the target of a link the user can ask the system to show the source of that link.

To show the source of a link the user should do the following:

1. Activate the target link's container.
2. Choose **Show links** from the **View** menu.
3. Double-click on the arrow in the highlighting of the target of a link.

Or:

1. Choose a **Link Target** or **Link source** from the Properties notebook.

Or:

1. Double-click on an icon of a Link Target.

Or:

1. For a Link Target shown as a frame, double-click the background.

If the source of the link is the same document as the target, the system scrolls the window to the source of the link and selects it. If the source is in a different document, the system opens the document, scrolls to the source of the link, and selects it. The source link border will become visible.

Hide Links

To hide links the user should do the following:

1. Activate the container of the links.
2. Choose **Hide links** from a **View** or pop-up menu.

Or:

1. Click outside the link.

Note: The second method works only if the border was shown as a side-effect of insertion or selection within a link.

When the user hides a link by either method, the link highlighting disappears. There is no visible indication that an item is linked.

Selecting Linked Parts or Data

To select a link the user should do the following:

1. Choose **Show links** from the **View** menu.
2. Click on the border of the link highlighting.

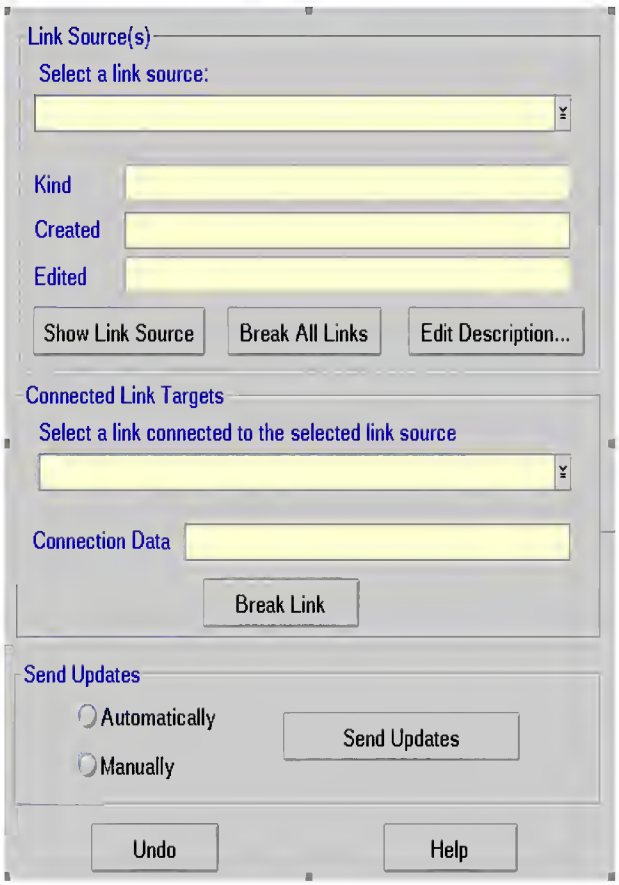
Linked parts and data can be selected just as any other parts or data. Note that links may be nested. That is, a source link might completely contain a source or target link, or source links might overlap. In this case, the ultimate target link does not show the nested links, because only the content, not the links themselves, is propagated through the link.

Showing Link Properties

To show link properties the user should do the following:

1. Select the part that contains the link.
2. Select **Properties** from the pop-up menu.

The Properties pages in the following figures are added to the Properties notebook of the part that contains the link. Additional properties pages can be added. For example, at a Link Target you might want to allow local properties for Fonts that override the settings in the source part.



Link Source Properties Notebook page

The properties for the **Link Source** are:

Select a Link Source

Allows the user to choose a link source. As the link sources change, the data presented in **Link Source**, **Connected Links**, and **Send Updates**, are updated.

Kind

Shows the part kind used for the link source. This field is scrollable and read-only.

Created

Shows the date and time the link source was created. This field is scrollable and read-only.

Edited

A read-only field showing the date and time the link was last edited.

Show Link Source

This push button allows the user to see where the link source is within a document. This push button is grayed if the document resides on a different machine.

Break All Links

If the user selects this push button, a confirmation dialog box is displayed. If the user confirms this choice, the link source is grayed in the **Select a Link** field until the user exits the notebook, and then the link is broken.

Edit Description

Allows the user to edit the current selected description.

Connected Links

This section provides information about the link targets corresponding to the selected link source.

Connection Data

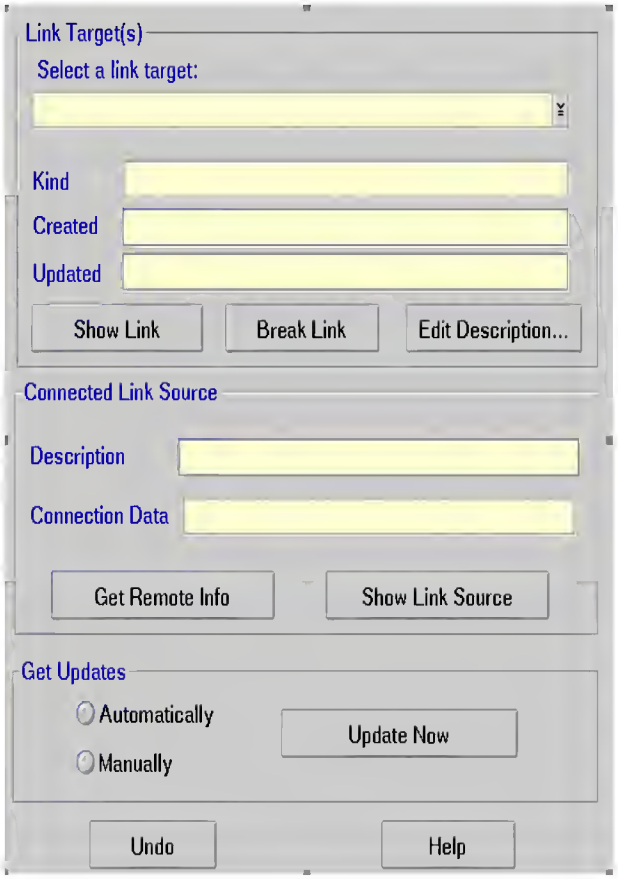
Displays the hostname and the document name of a link target.

Break Link

This push button causes a dialog box to be displayed. If the user confirms this action, the link is grayed in the **Connected Links** field until the user exits the notebook. Once the notebook is closed, the link between the selected link source and the selected link is broken.

Send Updates

A two-valued radio button (**Automatically**, **Manually**) which specifies when to propagate changes. **Automatically** means changes to the source will be propogated to the target whenever the user changes the source part. **Manually** means changes are propogated only when the user clicks the **Send Updates** push button; this inhibits the source from affecting anything unless the user explicitly directs it to do so, effectively disabling the link. The default is **Automatically**.



Link Target Properties Notebook page

The properties for the **Link Target** are:

Select a Link Target

This field shows a list of link targets in the part. As a link targets are selected, the data about the selected link target is reflected in the fields on this properties page.

Kind

Shows the user the part kind used for the link. This field is scrollable and read-only.

Created

Shows the user the date and time the link was created. This field is scrollable and read-only.

Updated

A read-only field showing the date and time the link was last updated.

Show Link

Shows the user where the link is in the part.

Break Link

This push button allows the user to break the selected link. If the user confirms the **Break link** selection, the link is broken after the properties notebook is closed.

Edit Description

Allows the user to edit the current selected description.

Connected Link Source

This section provides read-only text about the link source corresponding to the selected link.

Get Remote Data

A push button used to get the data about the remote link source. If the link source is local, the push button is grayed.

Show Link Source

The push button allows the user to find the link source within a document.

Get Updates

Radio buttons (**Automatically**, **Manually**) specifying when changes received from the source cause the target to update. **Automatically** means the target will be updated as soon as the change is received. **Manually** means only when the user selects the **Update Now** choice; this inhibits the target from displaying changes to the source when it receives them, effectively disabling the link. The default is **Automatically**.

Update Now

A push button choice that causes the target to be updated with the latest contents of the source. This choice is grayed when **Get Updates** is set to **Automatically**.

Editing the Source of a Link

Users can edit link sources in exactly the same way as they edit unlinked data, regardless of whether link highlighting is on or off, with no restrictions whatsoever. No warning message is posted when the source of a link is edited.

When a link source is modified, it propagates its changes to all of its targets, as specified by its Link Properties page described previously.

Editing the Target of a Link

The target of a link can be deleted, but it cannot be edited. Any attempt to edit the contents results in a message stating that a link target is being modified and asks if the user wants to see the source of the link for editing. If the user responds with an affirmation, the source of the link is displayed and is selected.

While the target of a link cannot be edited, it can be selected so that it can be copied or its properties changed. For example, if a bar chart is linked in from somewhere, the user can select one of the bars and change the color. If the link is updated, changing the content of the chart, the new color is preserved if possible.(**) Invalid choices for target operations will be disabled.

When the target of a link in a text part is active there is no persistent I-beam cursor. This is to indicate to users that they cannot edit the data. In addition, if the active part provides an information line, it should inform the user when a target link has been selected.

Updating a Link

To update a link the user should do the following:

1. Open the Properties notebook.
2. Choose **Update Now** for a Link Target, or **Send Updates** for a Link Source, in the Properties notebook.

This action causes data to be propagated (from the source end) or displayed (at the target end) immediately.

The following table describes the outcome of actions based on the settings in the **Link Source** and **Link Target** Properties pages.

```
 ÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜæÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜæÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜå
é SOURCE                 é TARGET                 é ACTION                 é
ËÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÆÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÆÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜÜë
é Automatically          é Manually               é Always updated         é
```

```
ËüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Automatically          é Manually              é Press Update now      é
ËüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Manually              é Automatically         é Press Send update     é
ËüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüÆüüüüüüüüüüüüüüüüüüüüüüüüüüüüë
é Manually              é Manually              é Press both            é
 üüüüüüüüüüüüüüüüüüüüüüüüüüüü üüüüüüüüüüüüüüüüüüüüüüüüüüüü üüüüüüüüüüüüüüüüüüüüüüüüüüüüä
```

Moving a Link

The move link action is the same as any other **Move** action throughout OpenDoc.

When the user moves a link, OpenDoc attempts to preserve the link. For example, if a source and target link are moved within the same document, or to a different document, OpenDoc preserves these links. However, if a single source or target link is moved to a different document, OpenDoc breaks that link(**). If the user attempts to do this, a warning message displays explaining that the link will be broken and a new one must be created if linking is desired.

In addition, some parts might not allow links to be pasted into them. For example, if some text that contains a Link Target is copied to a plain text part, it is likely that the Link Target will not be maintained. In this case, the user would get a message warning them that the link will be broken. If the user creates a Link Target, only the link content at the time of the copy will be seen in the target.

Because the contents of parts can be displayed in more than one frame, there are some interesting situations to consider when copying or moving a link. When the user cuts content including the source or target of a link, the clipboard will contain the link content and a paste would attempt to reconnect the link. At paste time, the target part creates a new part of the same type as the part containing the original content, puts in the contents, and hooks the target links back to their sources.

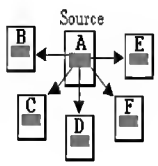
Copying a Link

If the user decides to copy a Link Source only (without its target), only its contents are copied to the clipboard. Its Link Target information is not copied because target links cannot have multiple sources. Links can be maintained only within documents. The new location may be in the same part as the original, or in a different part within the same document. Links are not copied from one document to another, except as source-target pairs.

Copying the source of a link does not alter the source or its links. They remain unchanged. The new copy initially has no links from it. The user can subsequently create links from the new copy, but these are independent of the old source and its links.

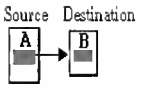
When a user copies a Link Target, or source-target pair, the linking information is copied along with the content.

Copying the Link Target *creates a new link to the same source* . Unlike the **Create link** operation, however, the **Paste as** dialog box does not appear, unless the **Paste as** choice is selected instead of the **Paste** choice. The links properties are set for the old target. As for a normal **Paste** operation, if the new target part cannot handle that data kind, **Embed** is automatically turned on, and the data is embedded in a part at the target. If the target does not allow embedding of that type of part, it does not enable the **Paste** and **Paste as** menu choices and it does not accept a drop.



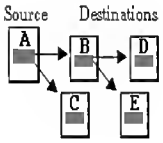
Single Target

The result is another link to the same source.



Multiple Targets

Not a chained link.



Chained Link, not available in OpenDoc

When only one source is involved, OpenDoc will not have chained dependencies between targets. However, because target links can be nested within different source links, it is possible to create dependencies between links similar to Chained Link, not available in OpenDoc., but involving a distinct source link at each stage.

Copying both the source and target of a link creates an entirely new link between the copy of the source and the copy of the target. The new link has no relationship to the original source or target or any of their links. If **Paste as** is selected, the user can create a new link to the same source, just like copying the target of a link. But **Paste as** also lets the user control kind, embedding, and appearance of the new link at the new target. Therefore, this is the recommended way to create multiple links from a single source. The same Link Source might appear quite differently at the different targets. See the *OpenDoc Progamming Guide* for more information about multiple links.

Deleting a Link

A link can be broken either by explicit user actions intended to break links, or as a side effect of another user action which links cannot survive.

To break a link the user can do the following:

1. Open the **Link Target** Properties notebook page.
2. Choose **Break link**.

Or:

1. Select a Link Target.
2. Choose **Break link** from the target containers' **Edit** menu.

Or:

1. Select the source or target of a link or both.
2. Choose **Cut** from the **Edit** or pop-up menu, or press the **Del** key.

When the user breaks a link, the actual content of the source and target does not change. However, subsequent changes to the source are not reflected in the target, since each target is now just an ordinary copy.

If the user deletes the source of a link, the links between it and all its targets are deleted. All targets now become disconnected copies with the content they had from the last update.

If the user deletes the target of a link, just the single link between it and the source is deleted. The source does not change.

The user can reverse the deletion of a link by selecting **Undo** in the **Edit** menu.

Scripting

Scripting allows customization of the interface. It is expected that most users will experience scripting indirectly, by using documents created from stationary that was created by third-party developers to provide extra functionality. For example, a user might create a new medical form for a patient and be unaware that data validation is performed by a script.

Supporting Scripting

Any part can have an associated script, either with the part as a whole or with some intrinsic content in the part. Some examples where scripts might be attached include: text fields, menu choices, push buttons, radio buttons, other controls and graphic objects.

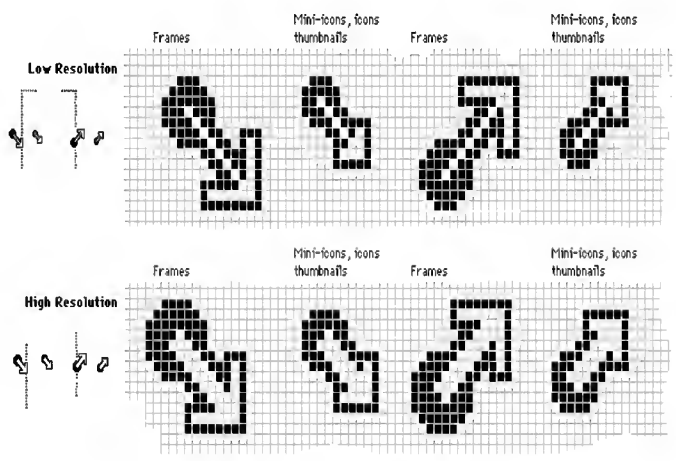
The user interface for attaching scripts to a part is defined by each part.

Appendix

This appendix details the various visuals used in OpenDoc for Link graphic details, Frame states, Crop indicators and Pointers.

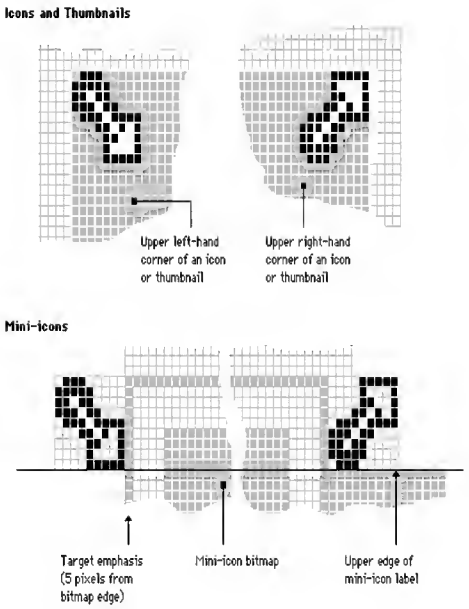
Link Graphic Details

The following figure shows the link graphic that should be used for frames, icons, and thumbnails. Note that different graphics should be used depending on the display resolution.



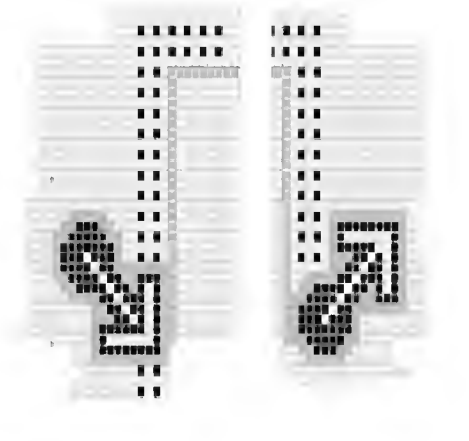
Pixel View of Link Graphic

The following figure shows how the link graphic is attached to icons and thumbnails. The picture on the right shows a source link, and the one on the left a target link.

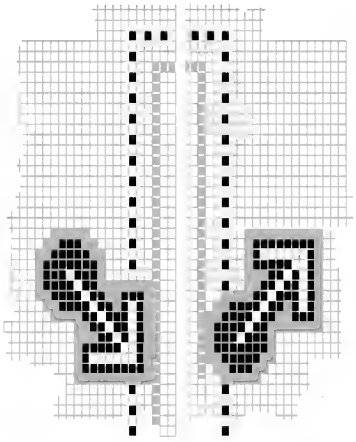


Link graphic position for icons, thumbnails, and mini-icons

The graphic indicating a link should be added to the active or selected border or to the dotted line that indicates the link boundaries at the bottom of the first quarter of the part's height, starting from the top. This position minimizes the risk of conflict with resize handles. The gray border around the graphic indicates a one-pixel clearance. Position of Link Graphic (Inactive Part) - Low Resolution. and Position of Link Graphic (Active or Selected part) - High Resolution. show the position of the link graphic in detail.

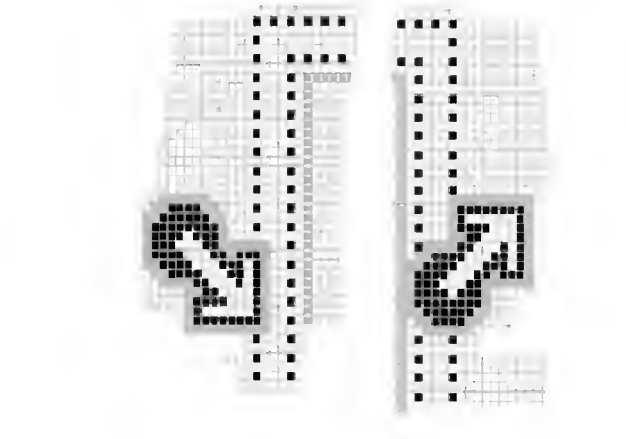


Position of Link Graphic (Active or Selected Part) - Low Resolution

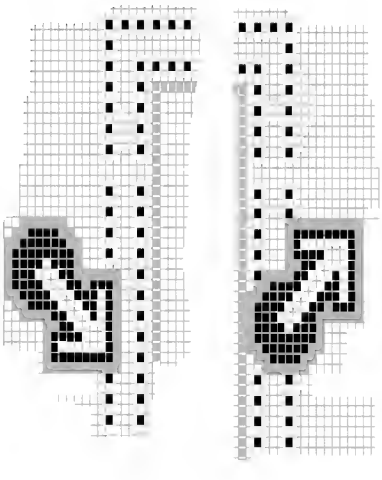


Link emphasis border for parts, not-selected state.

Position of Link Graphic (Inactive Part) - Low Resolution



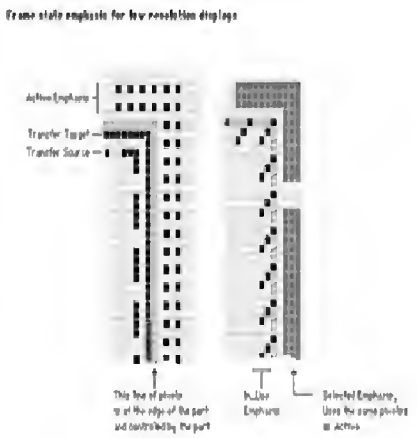
Position of Link Graphic (Active or Selected part) - High Resolution



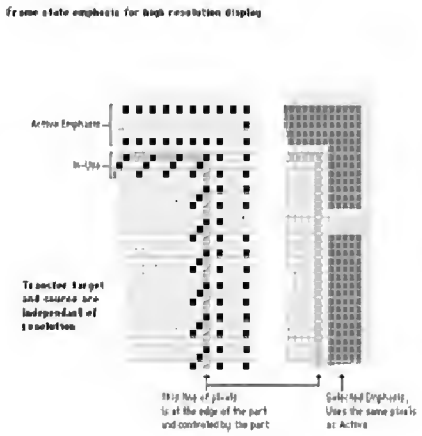
Position of Link Graphic (Inactive Part) - High Resolution

Active and Selected Frame Emphasis States

The following figures show the pixel-wise view of selected and active frame borders. Note that if an active frame cannot be sized it should not have a gap in the border as shown below.



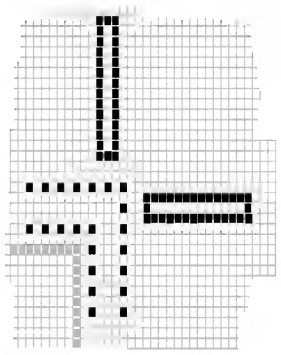
Frame Borders for Low-resolution Displays



Frame Borders for High-resolution Displays

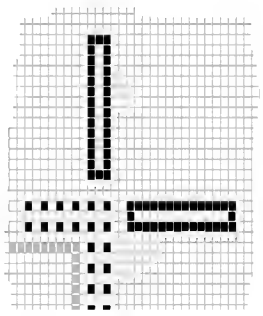
Crop Indicators

The following figures show the positioning of crop-mark indicators.



Positionning of crop indicators
for high resolution displays

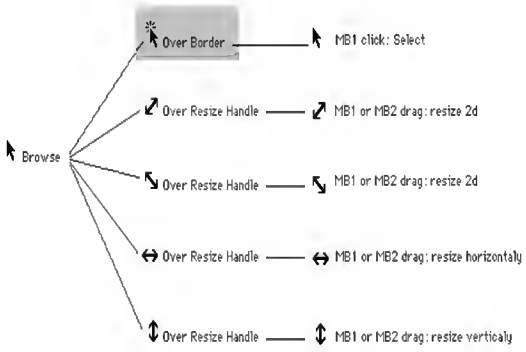
Crop Indicators for High-resolution Displays



Crop Indicators for Low-resolution Displays

Pointer Graphics

The following figure shows the pointers to be used in OpenDoc. The pointers shown in gray are new to OS/2.



Pointers used for OpenDoc

Notices

Edition Notices

First Edition (January 1996)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM reseller or IBM marketing representative.

Copyright Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Department RM1A, 1000 N.W. 51st Street, Boca Raton, FL 33431, U.S.A. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:
AIX
Common User Access
CUA
IBM
OS/2
Personal System/2
Presentation Manager
System Applications Architecture
SAA
Workplace Shell

The following terms are trademarks of other companies:

Apple	Apple Computer, Inc.
C++	American Telephone and Telegraph Company
CORBA	Object Management Group, Inc.
Helvetica	Linotype Company
Macintosh	Apple Computer, Inc.
Microsoft	Microsoft Corporation
OpenDoc	Apple Computer, Inc.
Palantino	Linotype Company
Postscript	Adobe Systems Incorporated
Windows	Microsoft Corporation

Other company, product, and service names, which may be denoted by a double asterisk (ì**), may be trademarks or service marks of others.

Glossary

This glossary defines many of the terms used in this book. It includes terms and definitions from the *IBM Dictionary of Computing* , as well as terms specific to the OS/2 operating system and the Presentation Manager. It is not a complete glossary for the entire OS/2 operating system; nor is it a complete dictionary of computer terms.

Other primary sources for these definitions are:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyrighted 1990 by the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. These definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* , developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

Glossary Listing

Select a starting letter of glossary terms:

Glossary - A

accelerator
In Common User Access architecture, a key or combination of keys that invokes an application-defined function.

accelerator table
A table used to define which key strokes are treated as *accelerators* and the commands they are translated into.

access mode
The manner in which an application gains access to a file it has opened. Examples of access modes are read-only, write-only, and read/write.

access permission
All access rights that a user has regarding an object. (I)

action
One of a set of defined tasks that a computer performs. Users request the application to perform an action in several ways, such as typing a command, pressing a function key, or selecting the action name from an action bar or menu.

action bar
In Common User Access architecture, the area at the top of a window that contains choices that give a user access to actions available in

that window.

action data
Information stored in the undo object's action history that allows a part to reverse the effects of an undoable action.

action history
The cumulative set of reversible actions available at any one time, maintained by the undo object.

action subhistory
A subset of action data added to the undo object s action history by a part in a modal state. The part can then remove the subhistory from the action history without affecting earlier actions.

action type
A constant that defines whether an undoable action is a single-stage action (such as a cut) or part of a two-stage action (such as a drag-move).

activate
To make ready to receive the selection focus. A frame is activated when a mouse-down event occurs within it. On most platforms, a window is activated when it is to the front, or when the cursor passes over it.

active frame
The frame that has the selection focus and usually the keyboard focus. Editing takes place in the active frame; the selection or insertion point is displayed within the frame.

active part
A part may be active or inactive. Being active means that the part contains the selection (or insertion cursor). For a part to be active, its contents must be visible, i.e., it must be displayed in a window or frame. Normally the active part receives commands and keyboard events, and its frame border, menu, palettes, and other user interface techniques are displayed.

One part is active at a time. A part is made active by user actions such as clicking the mouse in its contents. When a part is activated, the previously active part, if any, becomes inactive. The frame border, menu, palettes, etc. of inactive parts are not displayed. Being inactive does not mean that a part isn't running. Parts may execute asynchronously whether they are active or inactive, even if they are displayed as icons.

A part may not receive keyboard events nor put up its own menu or other user interface techniques when it is inactive.

active program
A program currently running on the computer. An active program can be interactive (running and receiving input from the user) or noninteractive (running but not receiving input from the user). See also *interactive program* and *noninteractive program* .

active shape
A shape that describes the portion of a facet within which a part expects to receive user events. If, for example, an embedded part's used shape and active shape are identical, the containing part both draws and accepts events in the unused areas within the embedded part's frame.

active window
The window with which the user is currently interacting.

American National Standard Code for Information Interchange
The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

Note: IBM has defined an extension to ASCII code (characters 128-255).

ancestor
See *superclass* .

anchor point
A point in a window used by a program designer or by a window manager to position a subsequently appearing window.

ANSI
American National Standards Institute.

APA
All points addressable.

API
Application programming interface.

application
A collection of software components used to perform specific types of user-oriented work on a computer; for example, a payroll application, an airline reservation application, a network application. See also *conventional application* .

application object
In Advanced Common User Access architecture, a form that an application provides for a user; for example, a spreadsheet form.

Contrast with *user object* .

application programming interface (API)
A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

application result handler
A result handler that is associated with a particular application. Compare with system result handler.

application-modal
Pertaining to a message box or dialog box for which processing must be completed before further interaction with any other window owned by the same application may take place.

arbitrator
An OpenDoc object that manages negotiation among parts about ownership of shared resources. Examples of such resources are the menu focus, selection focus, keystroke focus, and the serial ports.

area
In computer graphics, a filled shape such as a solid rectangle.

ASCII
American National Standard Code for Information Interchange.

ASCIIZ
A string of ASCII characters that is terminated with a byte containing the value 0.

aspect ratio
In computer graphics, the width-to-height ratio of an area, symbol, or shape.

asynchronous (ASYNC)
(1) Pertaining to two or more processes that do not depend upon the occurrence of specific events such as common timing signals. (T) (2) Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions. See also *synchronous* .

atom
A constant that represents a string. As soon as a string has been defined as an atom, the atom can be used in place of the string to save space. Strings are associated with their respective atoms in an *atom table* . See also *integer atom* .

atom table
A table used to relate *atoms* with the strings that they represent. Also in the table is the mechanism by which the presence of a string can be checked.

atomic operation
An operation that completes its work on an object before another operation can be performed on the same object.

attribute
A characteristic or property that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also *graphics attributes* and *segment attributes* .

automatic link
In Information Presentation Facility (IPF), a link that begins a chain reaction at the primary window. When the user selects the primary window, an automatic link is activated to display secondary windows.

auxiliary storage unit
An extra storage unit that a part uses to store its contents. Compare main storage unit.

AVIO
Advanced Video Input/Output.

Glossary - B

background
(1) In multiprogramming, the conditions under which low-priority programs are executed. Contrast with *foreground* . (2) An active session that is not currently displayed on the screen.

background color
The color in which the background of a graphic primitive is drawn.

background mix
An attribute that determines how the background of a graphic primitive is combined with the existing color of the graphics presentation space. Contrast with *mix* .

background program
In multiprogramming, a program that executes with a low priority. Contrast with *foreground program* .

base class
See superclass.

base menu bar
The menu bar that contains the menus shared by all parts in a document. The document shell installs the base menu bar; parts and their own menus and items.

base object
The object whose interface is extended by an extension object.

Bento
A document storage system, built on top of a platform's native file system, that allows for the creation, storage, and retrieval of compound documents. The OpenDoc storage system is based on Bento.

Bézier curve
(1) A mathematical technique of specifying smooth continuous lines and surfaces, which require a starting point and a finishing point with several intermediate points that influence or control the path of the linking curve. Named after Dr. P. Bézier. (2) (D of C) In the AIX Graphics Library, a cubic spline approximation to a set of four control points that passes through the first and fourth control points and that has a continuous slope where two spline segments meet. Named after Dr. P. Bézier.

bias transform
A transform that is applied to measurements in a part's coordinate system to change them into platform-normal coordinates.

binding
(1) In programming, an association between a variable and a value for that variable that holds within a defined scope. The scope may be that of a rule, a function call or a procedure invocation. (2) In OpenDoc, the process of selecting an executable code module based on type information.

bit map
A representation in memory of the data displayed on an APA device, usually the screen.

block
(1) A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or logical records. (T) (2) To record data in a block. (3) A collection of contiguous records recorded as a unit. Blocks are separated by interblock gaps and each block may contain one or more records. (A)

block device
A storage device that performs I/O operations on blocks of data called *sectors* . Data on block devices can be randomly accessed. Block devices are designated by a drive letter (for example, **C:**).

blocking mode
A condition set by an application that determines when its threads might block. For example, an application might set the Pipemode parameter for the DosCreateNPipe function so that its threads perform I/O operations to the named pipe block when no data is available.

border
(1) A visual indication (for example, a separator line or a background color) of the boundaries of a window. (2) For OpenDoc, see *frame border* .

boundary determination
An operation used to compute the size of the smallest rectangle that encloses a graphics object on the screen.

boundary objects
The elements, specified in a range descriptor record, that identify the beginning and end of the range. See also *range descriptor record* .

breakpoint
(1) A point in a computer program where execution may be halted. A breakpoint is usually at the beginning of an instruction where halts, caused by external intervention, are convenient for resuming execution. (T) (2) A place in a program, specified by a command or a condition, where the system halts execution and gives control to the workstation user or to a specified program.

buffer
(1) A portion of storage used to hold input or output data temporarily. (2) To allocate and schedule the use of buffers. (A)

bundled frame
A frame which has the Bundled property set. The system treats a bundled frame as a single atomic object. Its contents cannot be selected. Any mouse click inside the frames contents selects the entire frame. This allows the frame to be easily manipulated as a whole.

button
A mechanism used to request or initiate an action.

byte pipe
Pipes that handle data as byte streams. All unnamed pipes are byte pipes. Named pipes can be byte pipes or message pipes. See *byte stream* .

byte stream
Data that consists of an unbroken stream of bytes.

Glossary - C

cache
A high-speed buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

cached micro presentation space
A presentation space from a Presentation-Manager-owned store of micro presentation spaces. It can be used for drawing to a window only, and must be returned to the store when the task is complete.

CAD
Computer-Aided Design.

call
(1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) To transfer control to a procedure, program, routine, or subroutine.

calling sequence
A sequence of instructions together with any associated data necessary to execute a call. (T)

Cancel
An action that removes the current window or menu without processing it, and returns the previous window.

canvas
The platform-specific drawing environment on which frames are laid out. Each window or printing device has one drawing canvas. See also static canvas and dynamic canvas. See *drawing canvas* .

cascaded menu
In the OS/2 operating system, a menu that appears when the arrow to the right of a cascading choice is selected. It contains a set of choices that are related to the cascading choice. Cascaded menus are used to reduce the length of a menu. See also *cascading choice* .

cascading choice
In Common User Access architecture, a choice in a menu that, when selected, produces a cascaded menu containing other choices. An arrow ( ) appears to the right of the cascading choice.

CASE statement
In PM programming, provides the body of a window procedure. There is usually one CASE statement for each message type supported by an application.

category
See *part category* .

CGA
Color graphics adapter.

chained list
A list in which the data elements may be dispersed but in which each data element contains information for locating the next. (T)
Synonymous with *linked list* .

character
A letter, digit, or other symbol.

character box
In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single character from a character set. See also *character mode* . Contrast with *character cell* .

character cell
The physical, rectangular space in which any single character is displayed on a screen or printer device. Position is addressed by row and column coordinates. Contrast with *character box* .

character code
The means of addressing a character in a character set, sometimes called *code point* .

character device
A device that performs I/O operations on one character at a time. Because character devices view data as a stream of bytes, character-device data cannot be randomly accessed. Character devices include the keyboard, mouse, and printer, and are referred to by name.

character mode
A mode that, in conjunction with the font type, determines the extent to which graphics characters are affected by the character box, shear, and angle attributes.

character set
(1) An ordered set of unique representations called characters; for example, the 26 letters of English alphabet, Boolean 0 and 1, the set of symbols in the Morse code, and the 128 ASCII characters. (A) (2) All the valid characters for a programming language or for a computer system. (3) A group of characters used for a specific reason; for example, the set of characters a printer can print.

check box
In Advanced Common User Access architecture, a square box with associated text that represents a choice. When a user selects a choice, an **X** appears in the check box to indicate that the choice is in effect. The user can clear the check box by selecting the choice again. Contrast with *radio button* .

check mark
(1) (D of C) In Advanced Common User Access architecture, a check symbol that shows that a choice is currently in effect. (2) The symbol that is used to indicate a selected item on a pull-down menu.

child class
See *subclass* .

child process
In the OS/2 operating system, a process started by another process, which is called the parent process. Contrast with *parent process* .

child window
A window that appears within the border of its parent window (either a primary window or another child window). When the parent window is resized, moved, or destroyed, the child window also is resized, moved, or destroyed; however, the child window can be moved or resized independently from the parent window, within the boundaries of the parent window. Contrast with *parent window* .

choice
(1) An option that can be selected. The choice can be presented as text, as a symbol (number or letter), or as an icon (a pictorial symbol). (2) (D of C) In Common User Access architecture, an item that a user can select.

chord
(1) To press more than one button on a pointing device while the pointer is within the limits that the user has specified for the operating environment. (2) (D of C) In graphics, a short line segment whose end points lie on a circle. Chords are a means for producing a circular image from straight lines. The higher the number of chords per circle, the smoother the circular image.

CI Labs
See Component Integration Laboratories.

class
In object-oriented design or programming, a group of objects that share a common definition and that therefore share common properties, operations, and behavior. Members of the group are called instances of the class.

class hierarchy
The structure by which classes are related through inheritance.

class method
In System Object Model, an action that can be performed on a class object. Synonymous with factory method.

class object
In System Object Model, the run-time implementation of a class.

class style
The set of properties that apply to every window in a window class.

client
(1) A functional unit that receives shared services from a server. (T) (2) A user, as in a client process that uses a named pipe or queue that is created and owned by a server process.

client application
An application that uses OSA events to request a service (for example, printing a list of files, checking the spelling of a list of words, or performing a numeric calculation) from another application (called a server application). These applications can reside on the same local computer or on remote computers connected to a network.

client area
The part of the window, inside the border, that is below the menu bar. It is the user's work space, where a user types information and selects choices from selection fields. In primary windows, it is where an application programmer presents the objects that a user works on.

client program
An application that creates and manipulates instances of classes.

client window
The window in which the application displays output and receives input. This window is located inside the frame window, under the window title bar and any menu bar, and within any scroll bars.

clip limits
The area of the paper that can be reached by a printer or plotter.

clip shape
The structure by which classes are related through inheritance.

clipboard
In Common User Access architecture, an area of computer memory, or storage, that temporarily holds data. Data in the clipboard is available to other applications.

clipboard focus
In OpenDoc, a designation of ownership of access to the clipboard. The part with the clipboard focus can read from and write to the clipboard.

clipping
In computer graphics, removing those parts of a display image that lie outside a given boundary. (I) (A)

clipping area
The area in which the window can paint.

clipping path
A clipping boundary in world-coordinate space.

clock tick
The minimum unit of time that the system tracks. If the system timer currently counts at a rate of X Hz, the system tracks the time every 1/X of a second. Also known as *time tick* .

clone
To copy an object and all its referenced objects. When you clone an object, that object plus all other objects to which there is a strong persistent reference in the cloned object are copied.

close
For a frame, to remove from memory but not from storage. A closed frame is not permanently removed from its document. Compare remove.

code page
An assignment of graphic characters and control-function meanings to all code points.

code point
(1) Synonym for *character code* . (2) (D of C) A 1-byte code representing one of 256 potential characters.

code segment
An executable section of programming code within a load module.

color dithering
See *dithering* .

color graphics adapter (CGA)
An adapter that simultaneously provides four colors and is supported by all IBM Personal Computer and Personal System/2 models.

command
The name and parameters associated with an action that a program can perform.

command area
An area composed of a command field prompt and a command entry field.

command entry field
An entry field in which users type commands.

command line
On a display screen, a display line, sometimes at the bottom of the screen, in which only commands can be entered.

command mode
A state of a system or device in which the user can enter commands.

command prompt
A field prompt showing the location of the command entry field in a panel.

Common Object Request Broker Architecture (CORBA)
A standard promulgated by the Object Management Group industry consortium for defining interactions among objects.

Common Programming Interface (CPI)
Definitions of those application development languages and services that have, or are intended to have, implementations on and a high degree of commonality across the SAA environments. One of the three SAA architectural areas. See also *Common User Access architecture* .

Common User Access (CUA) architecture
Guidelines for the dialog between a human and a workstation or terminal. One of the three SAA architectural areas. See also *Common Programming Interface* .

compile
To translate a program written in a higher-level

compiled script file
A script file with the file type 'scpt' that contains script data as a resource of type 'scpt'. Before executing the script in a compiled script file, a user must first open the script from an application such as Script Editor.

component
(1) Hardware or software that is part of a functional unit. A functional part of an operating system; for example, the scheduler or supervisor. (2) A set of modules that performs a major function within a system; for example, a compiler or a master scheduler. (3) A software product that functions in the OpenDoc environment. Part editors, part viewers, and services are examples of components. See also *application component, service component* .

Component Integration Laboratories (CI Labs)
A consortium of platform and application vendors that oversees the development and distribution of OpenDoc technology.

composite window
A window composed of other windows (such as a frame window, frame-control windows, and a client window) that are kept together as a unit and that interact with each other.

compound document
A single document containing multiple, heterogeneous data types, each created, presented and edited by its own software. A compound document is made up of *parts* .

computer-aided design (CAD)
The use of a computer to design or change a product, tool, or machine, such as using a computer for drafting or illustrating.

COM1, COM2, COM3
Character-device names reserved for serial ports 1 through 3.

CON
Character-device name reserved for the console keyboard and screen.

concrete class
A class designed to be instantiated. Compare with *abstract class* .

conditional cascaded menu
A pull-down menu associated with a menu item that has a cascade mini-push button beside it in an object's pop-up menu. The conditional cascaded menu is displayed when the user selects the mini-push button.

container
(1) In Common User Access architecture, an object that holds other objects. A folder is an example of a container object. (2) A holder of persistent data (documents); part of the OpenDoc container suite. (3) An OSA event object that contains another OSA event object. A container is specified in an object specifier record by a keyword-specified descriptor record with the keyword keyAEContainer. The keyword-specified descriptor record is usually another object specifier record. It can also be a null descriptor record, or it can be used much like a variable when the OSA Event Manager determines a range or performs a series of tests. The objects a container contains can be either elements or properties. (See also *OSA event object* , *element* , *object specifier record* , *property* , *container part,* , *folder* and *object* .

container hierarchy
The chain of containers that determine the location of one or more OSA event objects. See also *container* .

container part
A part that is capable of embedding other parts or links to other parts. Compare with *simple part* . See embedding part.

containing part
The part that immediately contains an embedded part. Each embedded part has one containing part; each containing part has one or more embedded parts.

container suite
A set of OpenDoc classes that implement persistent storage. The container suite consists of containers, documents, drafts, and storage units.

containment
A relationship between objects wherein an object of one class contains a reference to an object of another class. Compare with *inheritance* .

content
See *part content* .

content area
The potentially visible area of a part as viewed in a frame or window. If the content area is greater than the area of the frame or window, only a portion of the part can be viewed at a time.

content coordinate space
The coordinate space defined by applying the internal transform of a frame to a point in frame coordinate space.

content element
A data item that can be seen by the user and is presented by a part's content model. Content elements can be manipulated through the graphical or scripting interface to a part.

content extent
The vertical dimension of the content area of a part in a frame. Content extent is used to calculate bias transforms.

content model
The specification of a part's contents (the data types of its content elements) and its content operations (the actions that can be performed on it and the interactions among its content elements).

content object
A content element that can be represented as an object and thus accessed and manipulated through semantic events.

content operation
A user action that manipulates a content element.

content storage unit
The main storage unit of the Clipboard, drag-and-drop object, link source object, or link object.

contextual help
In Common User Access Architecture, help that gives specific information about the item the cursor is on. The help is contextual because it provides information about a specific item as it is currently being used. Contrast with *extended help* .

contiguous
Touching or joining at a common edge or boundary, for example, an unbroken consecutive series of storage locations.

control
In Advanced Common User Access architecture, a component of the user interface that allows a user to select choices or type information; for example, a check box, an entry field, a radio button.

control area
A storage area used by a computer program to hold control information. (I) (A)

Control Program
(1) The basic functions of the operating system, including DOS emulation and the support for keyboard, mouse, and video input/output. (2) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A)

control window
A window that is used as part of a composite window to perform simple input and output tasks. Radio buttons and check boxes are examples.

control word
An instruction within a document that identifies its parts or indicates how to format the document.

conventional application
An application that directly handles events, opens documents, and is wholly responsible for manipulating, storing, and retrieving all of the data in its documents. Compare with *application component* .

coordinate bias
The difference between a given coordinate system and platform-normal coordinates. Coordinate bias typically involves both a change in axis polarity and an offset.

coordinate space
A two-dimensional set of points used to generate output on a video display of printer.

Copy
A choice that places onto the clipboard, a copy of what the user has selected. See also *Cut* and *Paste* .

CORBA
See Common Object Request Broker Architecture.

correlation
The action of determining which element or object within a picture is at a given position on the display. This follows a *pick* operation.

coverpage window
A window in which the application's help information is displayed

CPI
Common Programming Interface.

critical extended attribute
An extended attribute that is necessary for the correct operation of the system or a particular application.

critical section
(1) In programming languages, a part of an asynchronous procedure that cannot be executed simultaneously with a certain part of another asynchronous procedure.

Note: Part of the other asynchronous procedure also is a critical section. (2) A section of code that is not reentrant; that is, code that can be executed by only one thread at a time.

CUA architecture
Common User Access architecture. The architecture that defines the appearance of the user interface.

current draft
The most recent draft of a document. Only the current draft can be edited.

current frame
During drawing, the frame that is being drawn or within which editing is occurring.

current position
In computer graphics, the position, in user coordinates, that becomes the starting point for the next graphics routine, if that routine does not explicitly specify a starting point.

cursor
A symbol displayed on the screen and associated with an input device. The cursor indicates where input from the device will be placed. Types of cursors include text cursors, graphics cursors, and selection cursors. Contrast with *pointer* and *input focus* . Registrar to find out whether OSA events that already exist or are under development can be adapted to the needs of your application.

customizable
Characteristic of a scriptable part that also defines content objects and operations for interface elements such as menus and buttons.

Cut
In Common User Access architecture, a choice that removes a selected object, or a part of an object, to the clipboard, usually compressing the space it occupied in a window. See also *Copy* and *Paste* .

Glossary - D

daisy chain
A method of device interconnection for determining interrupt priority by connecting the interrupt sources serially.

database extension
The interface between the Data Access Manager and a data server.

data segment
A nonexecutable section of a program module; that is, a section of a program that contains data definitions.

data server
An application that acts as an interface between a database extension on a personal computer and a data source, which can be on the personal computer or on a remote host computer. A data server can be a database server program that can provide an interface to a variety of different databases, or it can be the data source itself, such as an application program.

data transfer
The movement of data from one object to another by way of the clipboard or by direct manipulation.

DBCS
Double-byte character set.

DDE
Dynamic data exchange.

deadlock
(1) Unresolved contention for the use of a resource. (2) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by, or a response from, the other. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

debug
To detect, diagnose, and eliminate errors in programs. (T)

decipoint
In printing, one tenth of a point. There are 72 points in an inch.

default container
The outermost container in an application's container hierarchy; usually the application itself. See also *container hierarchy* .

default editor for kind
A user-specified choice of part editor to use with parts whose preferred editor is not present.

default procedure
A function provided by the Presentation Manager Interface that may be used to process standard messages from dialogs or windows.

default value
A value assumed when no value has been specified. Synonymous with assumed value. For example, in the graphics programming interface, the default line-type is 'solid'.

definition list
A type of list that pairs a term and its description.

delta
An application-defined threshold, or number of container items, from either end of the list.

derived class
See *subclass* .

descendant
See *child process* and *subclass* .

descriptive text
Text used in addition to a field prompt to give more information about a field.

Deselect all
A choice that cancels the selection of all of the objects that have been selected in that window.

desktop window
The window, corresponding to the physical device, against which all other types of windows are established.

destination content
The content at the destination of a link. It is a copy of the source content.

destination part
A part that displays, through a link, information that resides in another part. Compare with *source part* .

detached process
A background process that runs independent of the parent process.

detent
A point on a slider that represents an exact value to which a user can move the slider arm.

device context
A logical description of a data destination such as memory, metafile, display, printer, or plotter. See also *direct device context* , *information device context* , *memory device context* , *metafile device context* , *queued device context* , and *screen device context* .

device driver
A file that contains the code needed to attach and use a device such as a display, printer, or plotter.

device space
(1) Coordinate space in which graphics are assembled after all GPI transformations have been applied. Device space is defined in device-specific units. (2) (D of C) In computer graphics, a space defined by the complete set of addressable points of a display device. (A)

dialog
The interchange of information between a computer and its user through a sequence of requests by the user and the presentation of responses by the computer.

dialog box
In Advanced Common User Access architecture, a movable window, fixed in size, containing controls that a user uses to provide

information required by an application so that it can continue to process a user request. See also *message box, primary window, secondary window* . Also known as a *pop-up window* .

Dialog Box Editor
A *WYSIWYG* editor that creates dialog boxes for communicating with the application user.

dialog item
A component (for example, a menu or a button) of a dialog box. Dialog items are also used when creating dialog templates.

dialog procedure
A dialog window that is controlled by a window procedure. It is responsible for responding to all messages sent to the dialog window.

dialog tag language
A markup language used by the DTL compiler to create dialog objects.

dialog template
The definition of a dialog box, which contains details of its position, appearance, and window ID, and the window ID of each of its child windows.

direct parameter
The parameter in an OSA event that contains the data or object specifier record to be used by the server application. For example, a list of documents to be opened is specified in the direct parameter of the Open Documents event. See also *OSA event parameter* .

direct device context
A logical description of a data destination that is a device other than the screen (for example, a printer or plotter), and where the output is not to go through the spooler. Its purpose is to satisfy queries. See also *device context* .

direct manipulation
The user's ability to interact with an object by using the mouse, typically by dragging an object around on the Desktop and dropping it on other objects.

direct memory access (DMA)
A technique for moving data directly between main storage and peripheral equipment without requiring processing of the data by the processing unit.(T)

directory
A type of file containing the names and controlling information for other files or other directories.

dispatcher
The OpenDoc object that directs user events and semantic events to the correct part.

dispatch module
An OpenDoc object used by the dispatcher to dispatch events of a certain type to part editors.

display frame
A frame in which a part is displayed. A part's display frames are created by and embedded in its containing part. Compare embedded frame.

display-frames list
A part's list of all the frames in which it is displayed. If a part is displayed in only one frame, it has only one element in this list.

display point
Synonym for *pel* .

display property
A visual characteristic of a containing part, such as its text font, that it makes available for embedded parts to adopt. Embedded parts can adopt the display characteristics of their containing parts that they understand, thus giving a more uniform appearance to a set of parts. Display properties are stored as properties in a storage unit passed from containing part to embedded part.

Distributed SOM (DSOM)
Distributed System Object Model. A version of SOM that provides remote access to SOM objects in a transparent way that insulates client programmers from having to know the location or platform type where a target object will be instantiated. DSOM allows programmers to use the same object model independently of whether the objects they access are in the same process, in another process on the same machine, or across distributed networks.

dithering
(1) The process used in color displays whereby every other pel is set to one color, and the intermediate pels are set to another. Together they produce the effect of a third color at normal viewing distances. This process can only be used on solid areas of color; it does not work, for example, on narrow lines. (2) (D of C) In computer graphics, a technique of interleaving dark and light pixels so that the resulting image looks smoothly shaded when viewed from a distance.

DMA
Direct memory access.

document
In OpenDoc, a user-organized collection of parts, all stored together.

document part
A part whose purpose is to present information to people and allow that information to be edited, such as a text document or a graphics document. The preferred representation for parts contained inside them is as frames. The name of the Document menu may be changed to match the category of the part or a name associated with the category. For example, if you have a graphics document the Document menu bar choice could be change to Graphic.

document process
A thread of execution that runs the document shell program. The document process provides the interface between the operating system and part editors. It accepts events from the operating system, provides the address space into which parts are loaded, and provides access to the window system and other features.

document shell
A program that provides an environment for all the parts in a document. The shell maintains the major document global databases: storage, window state, arbitrator, and dispatcher. This code also provides basic document behavior such as document creation, opening, saving, printing, and closing. OpenDoc provides a default document shell for each platform.

document window
The window for a document part. A spreadsheet window is an example. Compare with *part window* .

double-byte character set (DBCS)
A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more characters than can be represented by 256 code points, require double-byte character sets. Since each character requires two bytes, the entering, displaying, and printing of DBCS characters requires hardware and software that can support DBCS.

draft
A revision phase of a document on its way to becoming the final version. A document may have one or more drafts. Drafts appear in the Drafts view of a document.

drag
In Common User Access, to use a pointing device to move an object; for example, clicking on a window border, and dragging it to make the window larger.

drag and drop
A facility of OpenDoc that allows users to move or copy data through direct manipulation.

drag-copy
A drag-and-drop operation in which the dragged data remains at the source, and a copy is inserted at the destination.

drag-move
A drag-and-drop operation in which the dragged data is deleted from the source and inserted at the destination.

dragging
(1) In computer graphics, moving an object on the display screen as if it were attached to the pointer. (2) (D of C) In computer graphics, moving one or more segments on a display surface by translating. (I) (A)

drawing canvas
The platform-specific drawing environment on which frames are laid out. Each window or printing device has one drawing canvas. See also *static canvas* and *dynamic canvas* .

drawing chain
See *segment chain* .

drop
To fix the position of an object that is being dragged, by releasing the select button of the pointing device. See also *drag* .

DSOM
Distributed System Object Model. A version of SOM that works transparently over a network.

DTL
Dialog tag language.

DTS
Direct-To-SOM.

dual-boot function
A feature of the OS/2 operating system that allows the user to start DOS from within the operating system, or an OS/2 session from within DOS.

duplex
Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex* .

dynamic canvas
A drawing canvas that can potentially be changed, such as a window, that can be scrolled or paged to display different portions of a part' data. Compare with *static canvas* .

dynamic data exchange (DDE)
A message protocol used to communicate between applications that share data. The protocol uses shared memory as the means of exchanging data between applications.

dynamic data formatting
A formatting procedure that enables you to incorporate text, bit maps or metafiles in an IPF window at execution time.

dynamic link library
A collection of executable programming code and data that is bound to an application at load time or run time, rather than during linking. The programming code and data in a dynamic link library can be shared by several applications simultaneously.

dynamic linking
The process of resolving external references in a program module at load time or run time rather than during linking.

dynamic segments
Graphics segments drawn in exclusive-OR mix mode so that they can be moved from one screen position to another without affecting the rest of the displayed picture.

dynamic storage
(1) A device that stores data in a manner that permits the data to move or vary with time such that the specified data is not always available for recovery. (A) (2) A storage in which the cells require repetitive application of control signals in order to retain stored data. Such repetitive application of the control signals is called a refresh operation. A dynamic storage may use static addressing or sensing circuits. (A) (3) See also *static storage* .

dynamic time slicing
Varies the size of the time slice depending on system load and paging activity.

dynamic-link module
A module that is linked at load time or run time.

Glossary - E

EBCDIC
Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters (9 bits including parity check), used for information interchange among data processing systems, data communications systems, and associated equipment.

edge-triggered
Pertaining to an event semaphore that is posted then reset before a waiting thread gets a chance to run. The semaphore is considered to be posted for the rest of that thread's waiting period; the thread does not have to wait for the semaphore to be posted again.

edit-in-place
See *in-place editing* .

editor of last resort
The part editor that displays any part for which there is no available part editor on the system. The editor of last resort typically displays a gray rectangle representing the part's frame.

editor properties
A notebook, accessed through the **Edit** menu, in which the user can view and change properties for the part editor of the currently active part.

EGA
Extended graphics adapter.

embed
One part is inserted into the contents of another part. The inserted part maintains its part identity. The receiving part decides layout issues such as whether the new part overlaps existing parts in its contents. Compare with *incorporate* .

embedded content
Content displayed in an embedded frame. A containing part editor does not directly manipulate embedded content. Compare with *intrinsic content* .

embedded frame
A frame within which an embedded part is displayed. The embedded frame itself is considered intrinsic content of the containing part; the part displayed within the frame is not.

embedded-frames list
A containing part's list of all the frames embedded within it.

embedded part
A part displayed in an embedded frame. The data for an embedded part is stored within the same draft as its containing part. An embedded part is copied during a duplication of its containing part. An embedded part may itself be a containing part, unless it is a nonembedding part.

embedding part
A part that is capable of embedding other parts within its content; that is, it is capable of being a containing part. Compare nonembedding part.

EMS
Expanded Memory Specification.

encapsulation
Hiding an object's implementation, that is, its private, internal data and methods. Private variables and methods are accessible only to the object that contains them.

entry field
In Common User Access architecture, an area where a user types information. Its boundaries are usually indicated. See also *selection field* .

entry panel
A defined panel type containing one or more entry fields and protected information such as headings, prompts, and explanatory text.

entry-field control
The component of a user interface that provides the means by which the application receives data entered by the user in an entry field. When it has the input focus, the entry field displays a flashing pointer at the position where the next typed character will go.

environment segment
The list of environment variables and their values for a process.

environment strings
ASCII text strings that define the value of environment variables.

environment variables
Variables that describe the execution environment of a process. These variables are named by the operating system or by the application. Environment variables named by the operating system are PATH, DPATH, INCLUDE, INIT, LIB, PROMPT, and TEMP. The values of environment variables are defined by the user in the CONFIG.SYS file, or by using the SET command at the OS/2 command prompt.

error message
An indication that an error has been detected. (A)

event
See *user event* . Compare *semantic event* .

exclusive focus
A focus that can be owned by only one frame at a time. The selection focus, for example, is exclusive; the user can edit within only one frame at a time. Compare *non-exclusive focus* .

exclusive system semaphore
A system semaphore that can be modified only by threads within the same process.

executable file
(1) A file that contains programs or commands that perform operations or actions to be taken. (2) A collection of related data records that execute programs.

exit
To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on. (T) Repeated exit requests return the user to the point from which all functions provided to the system are accessible. Contrast with *cancel* .

expanded memory specification (EMS)
Enables DOS applications to access memory above the 1MB real mode addressing limit.

extended attribute
An additional piece of information about a file object, such as its data format or category. It consists of a name and a value. A file object may have more than one extended attribute associated with it.

extended help
In Common User Access architecture, a help action that provides information about the contents of the application window from which a user requested help. Contrast with *contextual help* .

extended-choice selection
A mode that allows the user to select more than one item from a window. Not all windows allow extended choice selection. Contrast with *multiple-choice selection* .

extension
An OpenDoc object that extends the programming interface of another OpenDoc object. Part editors, for example, can provide additional interfaces through extensions. An object class that duplicates all the characteristics of an object class of the same name and adds some of its own. Like a word in a dictionary, a single object class ID can have several related definitions.

extent
Continuous space on a disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

externalize
For a part or other OpenDoc object, to transform its in-memory representation into a persistent form in a storage unit. Compare *internalize* .

external link
In Information Presentation Facility, a link that connects external online document files.

external transform
A transform that is applied to a facet to position, scale, or otherwise transform the facet and the image drawn within it. Compare with *internal transform* .

extracted draft
A draft that is extracted from a document into a new document.

Glossary - F

facet
An object that describes where a frame is displayed on a canvas.

family-mode application
An application program that can run in the OS/2 environment and in the DOS environment; however, it cannot take advantage of many of the OS/2-mode facilities, such as multitasking, interprocess communication, and dynamic linking.

FAT
File allocation table.

FEA
Full extended attribute.

fidelity
The faithfulness of translation attained (or attainable) between data of different part kinds. For a given part kind, other part kinds are ranked in fidelity by the level at which their editors can translate its data without loss.

field-level help
Information specific to the field on which the cursor is positioned. This help function is "contextual" because it provides information about a specific item as it is currently used; the information is dependent upon the context within the work session.

FIFO
First-in-first-out. (A)

file
A named set of records stored or processed as a unit. (T)

file allocation table (FAT)
In IBM personal computers, a table used by the operating system to allocate space on a disk for a file, and to locate and chain together parts of the file that may be scattered on different sectors so that the file can be used in a random or sequential manner.

file attribute
Any of the attributes that describe the characteristics of a file.

file specification
The full identifier for a file, which includes its drive designation, path, file name, and extension.

file system
The combination of software and hardware that supports storing information on a storage device.

file system driver (FSD)
A program that manages file I\O and controls the format of information on the storage media.

fillet
A curve that is tangential to the end points of two adjoining lines. See also *polyfillet* .

filtering
An application process that changes the order of data in a queue.

first-in-first-out (FIFO)
A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

focus
A designation of ownership of a shared resource such as menus, selection, keystrokes, and serial ports. The part that owns a focus has use of that shared resource.

focus module
An OpenDoc object used by the arbitrator to assign an owner or owners to a given focus type.

focus set
A group of foci requested as a unit.

folder
A container used to organize objects.

font
A particular size and style of typeface that contains definitions of character sets, marker sets, and pattern sets.

Font Editor
A utility program provided with the IBM Developers Toolkit that enables the design and creation of new fonts.

foreground program
(1) The program with which the user is currently interacting. Also known as *interactive program* . Contrast with *background program* . (2) (D of C) In multiprogramming, a high-priority program.

frame
A representation for a part which displays the parts contents. It is the normal representation when a part is in a document window. The other possible representation is an icon.

frame border
A visible representation for a frame's shape. The border shows how clipping will occur and guides the positioning of contents inside the frame.

The border of a frame is displayed when the frame is active or selected; the border is invisible when the frame is inactive.

frame coordinate space
The coordinate space in which a part's frame shape, used shape, active shape, and clip shape are defined. Compare *content coordinate space* . See also *window coordinate space* , *canvas coordinate space* .

frame group
A set of a part's display frames that it designates as related, for purposes such as flowing content from one frame to another. Each frame group has its own group ID; frames within a frame group have a frame sequence.

frame negotiation
The process of adjusting the size and shape of an embedded frame. Embedded parts can request changes to their frames, but the containing parts control the changes that occur.

frame shape
The area available for displaying the frames contents. A frames shape is normally rectangular, although it need not be. For example, a button might have rounded corners. Some applications may provide direct manipulation of the shape, others may provide indirect manipulation via a menu or property sheet; while still others may provide no control over the shape. It is not required that the user be given any control over the shape.

frame styles
Standard window layouts provided by the Presentation Manager.

frame transform
The composite transform that converts from a part's frame coordinates to its canvas coordinates

frame view type
A view type in which all or a portion of a part's contents is displayed within a frame, the border of which is visible when the part is active or selected. Other possible view types for displaying a part include large icon, small icon, and thumbnail. Frame view type is sometimes called content view type.

FSD
File system driver.

full-duplex
Synonym for *duplex* .

full-screen application

An application that has complete control of the screen.

fully scriptable
Characteristic of a scriptable part in which semantic events can invoke any action a user might be able to perform.

function
(1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a call. (2) A set of related control statements that cause one or more programs to be performed.

function key
A key that causes a specified sequence of operations to be performed when it is pressed, for example, F1 and Alt-K.

function key area
The area at the bottom of a window that contains function key assignments such as F1=Help.

Glossary - G

global file-name character
Either a question mark (?) or an asterisk (*) used as a variable in a file name or file name extension when referring to a particular file or group of files.

glyph
A graphic symbol whose appearance conveys information.

GPI
Graphics programming interface.

graphic primitive
In computer graphics, a basic element, such as an arc or a line, that is not made up of smaller parts and that is used to create diagrams and pictures. See also *graphics segment* .

graphics
(1) A picture defined in terms of graphic primitives and graphics attributes. (2) (D of C) The making of charts and pictures. (3) Pertaining to charts, tables, and their creation. (4) See *computer graphics, coordinate graphics, fixed-image graphics, interactive graphics, passive graphics, raster graphics* .

graphics attributes
Attributes that apply to graphic primitives. Examples are color, line type, and shading-pattern definition. See also *segment attributes* .

graphics field
The clipping boundary that defines the visible part of the presentation-page contents.

graphics mode
One of several states of a display. The mode determines the resolution and color content of the screen.

graphics model space
The conceptual coordinate space in which a picture is constructed after any model transforms have been applied. Also known as *model space* .

Graphics programming interface
The formally defined programming language that is between an IBM graphics program and the user of the program.

graphics segment
A sequence of related graphic primitives and graphics attributes. See also *graphic primitive* .

graphics system
A specific drawing architecture. Some graphics systems (such as Display PostScript) are available on more than one platform; some platforms support more than one graphics system.

graying
The indication that a choice on a pull-down is unavailable.

group
A collection of logically connected controls. For example, the buttons controlling paper size for a printer could be called a group. See also *program group* .

group id
In OpenDoc, a number that identifies a frame group, assigned by the group's containing part.

Glossary - H

handle
(1) An identifier that represents an object, such as a device or window, to the Presentation Interface. (2) (D of C) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened.

hard error
An error condition on a network that requires either that the system be reconfigured or that the source of the error be removed before the system can resume reliable operation.

header
(1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message, such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

heading tags
A document element that enables information to be displayed in windows, and that controls entries in the contents window controls placement of push buttons in a window, and defines the shape and size of windows.

heap
An area of free storage available for dynamic allocation by an application. Its size varies according to the storage requirements of the application.

help function
(1) A function that provides information about a specific field, an application panel, or information about the help facility. (2) (D of C) One or more display images that describe how to use application software or how to do a system operation.

Help index
In Common User Access architecture, a help action that provides an index of the help information available for an application.

help panel
A panel with information to assist users that is displayed in response to a help request from the user.

help window
A Common User Access defined secondary window that displays information when the user requests help.

hidden file
An operating system file that is not displayed by a directory listing.

hide button
In the OS/2 operating system, a small, square button located in the right-hand corner of the title bar of a window that, when selected, removes from the screen all the windows associated with that window. Contrast with *maximize button* . See also *restore button* .

hierarchical inheritance
The relationship between parent and child classes. An object that is lower in the inheritance hierarchy than another object, inherits all the characteristics and behaviors of the objects above it in the hierarchy.

hierarchy
A tree of segments beginning with the root segment and proceeding downward to dependent segment types.

high-performance file system (HPFS)
In the OS/2 operating system, an installable file system that uses high-speed buffer storage, known as a cache, to provide fast access to large disk volumes. The file system also supports the coexistence of multiple, active file systems on a single personal computer, with the capability of multiple and different storage devices. File names used with the HPFS can have as many as 254 characters. OpenDoc uses HPFS.

hit testing
The means of identifying which window is associated with which input device event.

hook
A point in a system-defined function where an application can supply additional code that the system processes as though it were part of the function.

hook chain
A sequence of hook procedures that are "chained" together so that each event is passed, in turn, to each procedure in the chain.

hot part

A part, such as a control, that performs an action (like running a script), rather than activating itself, when it receives a mouse click.

hot spot
The part of the pointer that must touch an object before it can be selected. This is usually the tip of the pointer. Contrast with *action point* .

HPFS
high-performance file system.

hypergraphic link
A connection between one piece of information and another through the use of graphics.

hypertext
A way of presenting information online with connections between one piece of information and another, called *hypertext links* . See also *hypertext link* .

hypertext link
A connection between one piece of information and another.

Glossary - I

I/O operation
An input operation to, or output operation from a device attached to a computer.

I-beam pointer
A pointer that indicates an area, such as an entry field in which text can be edited.

icon
A representation for a part which displays a small picture along with a name. It is the normal representation when a part is in a desktop window. An icon can take the form of a regular icon, a small icon, or a thumbnail. The other possible representation is a frame.

icon area
In the Presentation Manager, the area at the bottom of the screen that is normally used to display the icons for minimized programs.

Icon Editor
The Presentation Manager-provided tool for creating icons.

identity transform
A transform that has no effect on points to which it is applied.

IDL
Interface Definition Language.

image font
A set of symbols, each of which is described in a rectangular array of pels. Some of the pels in the array are set to produce the image of one of the symbols. Contrast with *outline font* .

implied length
The definition of a specific length for a data type. An example of this is the Data Access Manager's typeInteger data type, which has a defined length of 4 bytes.

incorporate
To merge the data from one part into the contents of another part so that the merged data retains no separate identity as a part. Compare with *embed* .

indirect manipulation
Interaction with an object through choices and controls.

information device context
A logical description of a data destination other than the screen (for example, a printer or plotter), but where no output will occur. Its purpose is to satisfy queries. See also *device context* .

information panel
A defined panel type characterized by a body containing only protected information.

Information Presentation Facility (IPF)
A facility provided by the OS/2 operating system, by which application developers can produce online documentation and context-sensitive online help panels for their applications.

inheritance

The passing of class resources or attributes from a parent class downstream in the class hierarchy to a child class. The new class inherits all the data and methods of the parent class without having to redefine them.

in-place editing
Manipulation by a user of data in an embedded part without leaving the context of the document in which the part is displayed (for example, without opening a new window for the part).

input focus
(1) The area of a window where user interaction is possible using an input device, such as a mouse or the keyboard. (2) The position in the *active window* where a user's normal interaction with the keyboard will appear.

input router
An internal OS/2 process that removes messages from the system queue.

input/output control
A device-specific command that requests a function of a device driver.

insertion location descriptor record
A record of type typeInsertionLoc that consists of two keyword-specified descriptor records. The first is an object specifier record, and the data for the second is a constant that specifies the insertion location in relation to the OSA event object described by the object specifier record.

inside-out activation
A mode of user interaction in which a mouse click anywhere in a document activates the smallest possible enclosing frame and performs the appropriate selection action on the content element at the click location. OpenDoc uses inside-out selection. Compare outside-in activation.

inside-out selection
A mode of user interaction in which a mouse click anywhere in a document activates the smallest possible enclosing frame and performs the appropriate selection action on the content element at the click location. OpenDoc uses inside-out selection. Compare with *outside-in selection* .

installable file system (IFS)
A file system in which software is installed when the operating system is started.

instance
A single occurrence of an object class that has a particular behavior. See also *object* .

instantiate
(1) To make an instance of; to replicate. (2) In object-oriented programming, to represent a class abstraction with a concrete instance of the class.

instruction pointer
A pointer that provides addressability for a machine interface instruction in a program.

integer atom
An *atom* that represents a predefined system constant and carries no storage overhead. For example, names of window classes provided by Presentation Manager are expressed as integer atoms.

interactive graphics
Graphics that can be moved or manipulated by a user at a terminal.

interactive program
(1) A program that is running (active) and is ready to receive (or is receiving) input from a user. (2) A running program that can receive input from the keyboard or another input device. Compare with *active program* and contrast with *noninteractive program* .

Also known as a *foreground program* .

interchange file
A file containing data that can be sent from one application to another.

Interface Definition Language (IDL)
Language-neutral syntax created by IBM to describe the interface of classes that can be compiled by the SOM compiler.

internalize
For a part or other OpenDoc object, to transform its persistent form in a storage unit intto an appropriate in-memory representation. Compare *externalize* .

internal transform
A transform that positions, scales, or otherwise transforms the image of a part drawn within a frame. Compare with external transform.

interoperability
Access to an OpenDoc part or document from different platforms or with different software systems.

interpreter
A program that translates and executes each instruction of a high-level programming language before it translates and executes.

interprocess communication (IPC)
In the OS/2 operating system, the exchange of information between processes or threads through semaphores, pipes, queues, and shared memory.

intrinsic content
Data which is intrinsic to a particular type of part. For example, text parts contain characters. Graphics parts may contain lines and circles. Spreadsheet parts contain spreadsheet cells. Video parts contain digitized video. Sound parts contain digitized sound. Simulation parts contain executable code. And so on. The part developer determines what intrinsic contents a part may contain. See *part content* .

invalid shape
The area of a frame, facet, or canvas that needs redrawing. Update events cause redrawing of the invalid area.

invalidate
To mark an area of a canvas (or facet, or frame) as in need of redrawing.

invariant
An aspect of the internal state of an object that must be maintained for the object to behave properly according to its design.

IPC
Interprocess communication.

IPF
Information Presentation Facility.

IPF compiler
A text compiler that interpret tags in a source file and converts the information into the specified format.

IPF tag language
A markup language that provides the instructions for displaying online information.

Glossary - J

journal
A special-purpose file that is used to record changes made in the system.

Glossary - K

Kanji
A graphic character set used in Japanese ideographic alphabets.

KBD$
Character-device name reserved for the keyboard.

kernel
The part of an operating system that performs basic functions, such as allocating hardware resources.

kerning
The design of graphics characters so that their character boxes overlap. Used to space text proportionally.

keyboard accelerator
A keystroke that generates a command message for an application.

keyboard augmentation
A function that enables a user to press a keyboard key while pressing a mouse button.

keyboard focus
A temporary attribute of a window. The window that has a keyboard focus receives all keyboard input until the focus changes to a different window.

Keys help
In Common User Access architecture, a help action that provides a listing of the application keys and their assigned functions.

keystroke focus
A designation of ownership of keystroke events. The part whose frame has the keystroke focus receives keystroke events. See also s*election focus* .

keystroke focus frame
The frame to which keystroke events are to be sent.

kind
See *part kind* .

Glossary - L

label
In a graphics segment, an identifier of one or more elements that is used when editing the segment.

LAN
local area network.

large icon view type
A view type in which a part is represented by a 32 by 32-pixel bit map image. Other possible view types for displaying a part include small icon, thumbnail, and frame.

layout
The process of arranging frames and content elements in a document for drawing.

lazy drag
See *pickup and drop* .

lazy drag set
See *pickup set* .

lazy instantiation
The process of creating objects (such as embedded frames) in memory only when they are needed for display, such as when the user scrolls them into view. Lazy instantiation can help minimize the memory requirements of your parts.

link
A link is a relationship between two parts or pieces of part contents. A Link operation is like a Copy except that the link is updated every time the original changes. Links allow part contents to appear to be in two different places, even though it is only stored once. For instance, a document might link in a picture from another document, or a bar chart may display some data which is linked in from a row of a spreadsheet in the same document. Links may be within a part, between two parts in the same document, or between two parts in different documents. Both part contents and entire parts may be linked. Links are one way only. Linking is different from embedding or copying.

link source
The portion of a part's content area that represents the source of a link.

link specification
An object, placed on the Clipboard or in a drag-and-drop object, from which the source part (the part that placed the data) can construct a link if necessary.

link status
The link-related state (in a link source, in a link destination, or not in a link) of a frame.

linked list
Synonym for *chained list* .

linked part
A part (or a portion of a part's content data) that appears to the user to be embedded in one part, but it is actually embedded in a different part. Linked data is not copied when the link's containing part is duplicated; a new link is created instead.

list box
In Advanced Common User Access architecture, a control that contains scrollable choices from which a user can select one choice.

Note: In CUA architecture, this is a programmer term. The end user term is selection list.

list button
A button labeled with an underlined down-arrow that presents a list of valid objects or choices that can be selected for that field.

list panel
A defined panel type that displays a list of items from which users can select one or more choices and then specify one or more actions to work on those choices.

load
For a part editor, to transform the persistent form of a part in a draft into an appropriate in-memory representation, which can be a representation of the complete part or only a subset, depending on the current display requirements of the document. Compare with *save* .

local area network (LAN)
(1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T)

Note: A LAN does not use store and forward techniques. (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network.

lock
A serialization mechanism by means of which a resource is restricted for use by the holder of the lock.

Glossary - M

*

main window
The window that is positioned relative to the *desktop window* .

manipulation button
The button on a pointing device a user presses to directly manipulate an object.

map
(1) A set of values having a defined correspondence with the quantities or values of another set. (I) (A) (2) To establish a set of values having a defined correspondence with the quantities or values of another set. (I)

marker box
In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single marker from a marker set.

marker symbol
A symbol centered on a point. Graphs and charts can use marker symbols to indicate the plotted points.

marquee box
The rectangle that appears during a selection technique in which a user selects objects by drawing a box around them with a pointing device.

Master Help Index
In the OS/2 operating system, an alphabetic list of help topics related to using the operating system.

maximize
To enlarge a window to its largest possible size.

media window
The part of the physical device (display, printer, or plotter) on which a picture is presented.

member function
See *method* .

memory block
Part memory within a heap.

memory device context
A logical description of a data destination that is a memory bit map. See also *device context* .

menu
In Advanced Common User Access architecture, an extension of the menu bar that displays a list of choices available for a selected choice in the menu bar. After a user selects a choice in menu bar, the corresponding menu appears. Additional pop-up windows can appear from menu choices.

menu bar
In Advanced Common User Access architecture, the area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus.

menu button
The button on a pointing device that a user presses to view a pop-up menu associated with an object.

message
(1) In the Presentation Manager, a packet of data used for communication between the Presentation Manager interface and Presentation Manager applications (2) In a user interface, information not requested by users but presented to users by the computer in response to a user action or internal process. See also *semantic event* .

message box
(1) A dialog window predefined by the system and used as a simple interface for applications, without the necessity of creating dialog-template resources or dialog procedures. (2) (D of C) In Advanced Common User Access architecture, a type of window that shows messages to users. See also *dialog box, primary window, secondary window* .

message filter
The means of selecting which messages from a specific window will be handled by the application.

message interface
An OpenDoc object that provides an interface to allow parts to send messages (semantic events) to other parts, either in the same document or in other documents.

message queue
A sequenced collection of messages to be read by the application.

metafile
A file containing a series of attributes that set color, shape and size, usually of a picture or a drawing. Using a program that can interpret these attributes, a user can view the assembled image.

metafile device context
A logical description of a data destination that is a metafile, which is used for graphics interchange. See also *device context* .

metalanguage
A language used to specify another language. For example, data types can be described using a metalanguage so as to make the descriptions independent of any one computer language.

method
An function that manipulates the data of a particular class of objects.

method override
The replacement, by a child class, of the implementation of a method inherited from a parent and an ancestor class.

mickey
A unit of measurement for physical mouse motion whose value depends on the mouse device driver currently loaded.

micro presentation space
A graphics presentation space in which a restricted set of the GPI function calls is available.

minimize
To remove from the screen all windows associated with an application and replace them with an icon that represents the application.

mix
An attribute that determines how the foreground of a graphic primitive is combined with the existing color of graphics output. Also known as *foreground mix* . Contrast with *background mix* .

mixed character string
A string containing a mixture of one-byte and *Kanji* or Hangeul (two-byte) characters.

mnemonic
(1) A method of selecting an item on a pull-down by means of typing the highlighted letter in the menu item. (2) (D of C) In Advanced Common User Access architecture, usually a single character, within the text of a choice, identified by an underscore beneath the character. If all characters in a choice already serve as mnemonics for other choices, another character, placed in parentheses immediately following the choice, can be used. When a user types the mnemonic for a choice, the choice is either selected or the cursor is moved to that choice.

modal dialog box
In Advanced Common User Access architecture, a type of movable window, fixed in size, that requires a user to enter information before continuing to work in the application window from which it was displayed. Contrast with *modeless dialog box* . Also known as a *serial dialog box* . Contrast with *parallel dialog box* .

Note: In CUA architecture, this is a programmer term. The end user term is pop-up window.

modal focus
A designation of ownership of the right to display modal dialog boxes. A part displaying a modal dialog must first acquire the modal focus, so that other parts cannot do the same until the first part is finished.

model space

See *graphics model space* .

modeless dialog box
In Advanced Common User Access architecture, a type of movable window, fixed in size, that allows users to continue their dialog with the application without entering information in the dialog box. Also known as a *parallel dialog box* . Contrast with *modal dialog box* .

Note: In CUA architecture, this is a programmer term. The end user term is pop-up window.

module definition file
A file that describes the code segments within a load module. For example, it indicates whether a code segment is loadable before module execution begins (preload), or loadable only when referred to at run time (load-on-call).

monitor
A special use of a dispatch module, in which it is installed in order to be notified of events, but does not dispatch them.

monolithic application
See *conventional application* .

mouse
In CUA usage, a device that a user moves on a flat surface to position a pointer on the screen. It allows a user to select a choice o function to be performed or to perform operations on the screen, such as dragging or drawing lines from one position to another.

multiple-choice selection
In Basic Common User Access architecture, a type of field from which a user can select one or more choices or select none. See also *check box* . Contrast with *extended-choice selection* .

multiple-line entry field
In Advanced Common User Access architecture, a control into which a user types more than one line of information. See also *single-line entry field* .

multitasking
The concurrent processing of applications or parts of applications. A running application and its data are protected from other concurrently running applications.

Glossary - N

name space
An object consisting of a set of text strings used to identify kinds of objects or classes of behavior, for registration purposes. For example, OpenDoc uses name spaces to identify part kinds and categories for binding.

name-space manager
An OpenDoc object that creates and deletes *name spaces* .

national language support (NLS)
The modification or conversion of a United States English product to conform to the requirements of another language or country. This can include the enabling or retrofitting of a product and the translation of nomenclature, MRI, or documentation of a product.

nested list
A list that is contained within another list.

NLS
national language support.

nonembedding part
A part that cannot itself contain embedded parts; that is, it can never be a containing part. Compare with embedding part.

non-exclusive focus
A focus that can be owned by more than one frame at a time. OpenDoc supports the use of nonexclusive foci. Compare with *exclusive focus* .

noncritical extended attribute
An extended attribute that is not necessary for the function of an application.

nondestructive read
Reading that does not erase the data in the source location. (T)

noninteractive program
A running program that cannot receive input from the keyboard or other input device. Compare with *active program* , and contrast with

interactive program .

nonpersistent frame
A frame that exists as an object in memory, but has no storage unit and is not stored persistently.

nonretained graphics
Graphic primitives that are not remembered by the Presentation Manager interface when they have been drawn. Contrast with *retained graphics* .

null character (NUL)
(1) Character-device name reserved for a nonexistent (dummy) device. (2) (D of C) A control character that is used to accomplish media-fill or time-fill and that may be inserted into or removed from a sequence of characters without affecting the meaning of the sequence; however, the control of equipment or the format may be affected by this character. (I) (A)

Glossary - O

object
A programming entity, existing in memory at run time, that is an individual instantiation of a particular class.

object callback
A function called by the name resolver to allow your part to provide extra information needed for semantic-event object resolution.

object class ID
A four-character code, which can also be represented by a constant, that identifies an object class for an OSA event object. The object class ID for a primitive object class is the same as the four-character value of its descriptor type.

object class inheritance hierarchy
The hierarchy of subclasses and superclasses that determines which properties, elements, and OSA events object classes inherit from other object classes.

object-comparison function
An object callback function that compares an element to either another element or to a descriptor record and returns either TRUE or FALSE.

object-counting function
An object callback function that counts the number of elements of a specified class in a specified container, so that the OSA Event Manager can determine how many elements it must examine to find the element or elements that pass a test.

Object Interface Definition Language (OIDL)
Specification language used in SOM Version 1 for defining classes. Replaced by Interface Definition Language (IDL).

object linking and embedding (OLE)
An application protocol developed by Microsoft Corporation that allows objects created by one application to be linked to or embedded in objects created by another application.

object-marking function
An object callback function called repeatedly by the OSA Event Manager to mark specific OSA event objects. See also *marking callback functions* .

object model
A feature of OSA events that allows a part to define a hierarchical arrangement of content objects to represent the elements of the part's content.

object specifier
A designation of a content object within a part, used to determine the target of a semantic event. Object specifiers can be names ("blue rectangle") or logical designations ("word 1 of line 2 of embedded frame 3").

object window
A window that does not have a parent but which might have child windows. An object window cannot be presented on a device.

OIDL
Object Interface Definition Language.

OLE
See *Object Linking and Embedding* .

open
To start working with a file, directory, or other object.

Open Linking and Embedding of Objects (OLEO)
A technology that enables seamless interoperability between OpenDoc and Microsoft Corporation's Object Linking and Embedding (OLE) technology for interapplication communication. It allows OLE objects to function automatically as parts in OpenDoc documents, and OpenDoc parts to function automatically as OLE objects in OLE containers.

Open Scripting Architecture (OSA)
A mechanism based on the OSA Event Manager and the OSA Event Registry: Standard Suites that allows users to control multiple applications by means of scripts. The scripts can be written in any scripting language that supports the OSA.

OpenDoc
A multiplatform technology, implemented as a set of shared libraries, that uses component software to facilitate the construction and sharing of compound documents.

ordered list
Vertical arrangements of items, with each item in the list preceded by a number or letter.

outline font
A set of symbols, each of which is created as a series of lines and curves. Synonymous with *vector font* . Contrast with *image font* .

output area
An area of storage reserved for output. (A)

outside-in activation
A mode of user interaction in which a mouse click anywhere in a document activates the largest possible enclosing frame that is not already active. Compare inside-out activation.

outside-in selection
A mode of user interaction in which a mouse click anywhere in a document activates the largest possible enclosing frame that is not already active. compare *inside-out* se*lection* .

overlaid frame
An embedded frame that floats above the content (including other embedded frames) of its containing part, and thus need not engage in frame negotiation with the containing part.

override
To replace a method belonging to a superclass with a method of the same name in a subclass, in order to modify its behavior.

owner
For a canvas, the part that created the canvas and attached it to a facet. The owner is responsible for transferring the results of drawing on the canvas to its parent canvas.

owner window
A window into which specific events that occur in another (owned) window are reported.

ownership
The determination of how windows communicate using messages.

owning process
The process that owns the resources that might be shared with other processes.

Glossary - P

page
(1) A 4KB segment of contiguous physical memory. (2) (D of C) A defined unit of space on a storage medium.

page viewport
A boundary in device coordinates that defines the area of the output device in which graphics are to be displayed. The presentation-page contents are transformed automatically to the page viewport in device space.

paint
(1) The action of drawing or redrawing the contents of a window. (2) In computer graphics, to shade an area of a display image; for example, with crosshatching or color.

panel
In Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

panel area
An area within a panel that contains related information. The three major Common User Access-defined panel areas are the action bar,

the function key area, and the panel body.

panel area separator
In Basic Common User Access architecture, a solid, dashed, or blank line that provides a visual distinction between two adjacent areas of a panel.

panel body
The portion of a panel not occupied by the action bar, function key area, title or scroll bars. The panel body can contain protected information, selection fields, and entry fields. The layout and content of the panel body determine the panel type.

panel body area
See *client area* .

panel definition
A description of the contents and characteristics of a panel. A panel definition is the application developer's mechanism for predefining the format to be presented to users in a window.

panel ID
In Basic Common User Access architecture, a panel identifier, located in the upper-left corner of a panel. A user can choose whether to display the panel ID.

panel title
In Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

paper size
The size of paper, defined in either standard U.S. or European names (for example, A, B, A4), and measured in inches or millimeters respectively.

parallel dialog box
See *modeless dialog box* .

parameter list
A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

parent class
See *superclass* .

parent process
In the OS/2 operating system, a process that creates other processes. Contrast with *child process* .

parent window
In the OS/2 operating system, a window that creates a child window. The child window is drawn within the parent window. If the parent window is moved, resized, or destroyed, the child window also will be moved, resized, or destroyed. However, the child window can be moved and resized independently from the parent window, within the boundaries of the parent window. contrast with *child window* .

part
The fundamental building blocks in OpenDoc. They are the primitives with which people accomplish tasks. Parts replace today's monolithic applications. All parts have contents and properties. Parts may be represented on the display screen as either icons or frames. Every part is contained in some other part, either a document part or a desktop part. Synonymous with *document part* .

part category
A set of part kinds. A category is a general classification of a parts contents; for example, text, 2D graphics, 3D graphics, table, sound, video, animation, simulation, script. Developers decide on the categories. OpenDoc uses them for two purposes: (1) to determine the set of part editors that are applicable to a given part, and (2) to decide if it is appropriate to translate data during inter-part editing. Compare with *part kind* .

part container
See *container part* .

part content
In general, a part can contain two types of data: intrinsic contents and other parts. There is no requirement that parts be able to do contain other parts, and some won't be able to. But a key characteristic of OpenDoc is that if a part can contain one kind of part, it can contain all kinds of parts, since OpenDoc supplies a uniform wrapper for parts. (Contrast this with the small number of standard data types supported today, such as text, PICT, TIFF, etc.) Compare with *part editor* ; see also *part* .

part editor
Code that allows the user to modify the contents of a part. One of the main contributions of OpenDoc is giving users the ability to change part editors. This lets OpenDoc users employ their favorite editors for common types of data, such as text and graphics, regardless of how the data was created. See part viewer. Compare with *part content, part viewer* .

part ID
An identifier that uniquely names a part within the context of a document. This ID represents a storage unit ID within a particular draft of a document.

part info
(1) Part-specific data, of any type or size, used by a part editor to identify what should be displayed in a particular frame or facet and how it should be displayed. (2) Information about a given part that can be seen by the user and is displayed in a dialog box accessed through a menu command.

part kind
The format of a parts contents; for example, SurfWriter Text 3.0, SurfWriter Text 2.0, SurfPaint 1.2, SurfDraw 5.0. See *part category* .

part kind
A specific classification of the format of data handled by a part editor. A kind specifies the specific data format handled by, and possibly native to, a part editor. Kinds are meaningful to end-users and have designations such as such as 'MyEditor 2.0' or 'MyEditor 1.0'. Compare with *part category* .

part properties
Data which describe a part. The user may modify some properties, such as name, comments, and scripts. Other properties are set by the system, such as storage size, last modification date, and part type. Properties are displayed in the parts properties notebook. See also *property* .

part registry
The mechanism by which the document shell maps parts to part editors according to their part kind.

part table
A list of all the parts contained within a document and a list of associated data.

part viewer
Code that displays a parts contents. OpenDoc developers are urged to separate part viewers from part editors, so that viewers can be bundled with the contents when they are given to other people, e.g. via electronic mail. This allows the contents to at least be displayed, even if the user has not purchased the part editor. Compare with *part editor* .

part window
A window that displays an embedded part by itself, for easier viewing or editing. Any part that is embedded in another part can be opened up into its own part window. The part window is separate from and has a slightly different appearance than the *document window* that displays the entire document in which the part is embedded.

partition
(1) A fixed-size division of storage. (2) On an IBM personal computer fixed disk, one of four possible storage areas of variable size; one may be accessed by DOS, and each of the others may be assigned to another operating system.

Paste
A choice in the **Edit** pull-down that a user selects to move the contents of the clipboard into a preselected location. See also *Copy* and *Cut* .

Paste As
A choice in the **Edit** pull-down that a user selects to move the contents of the clipboard into a preselected location by means of a dialog box allowing the user to specify the format of the data. See also *Copy* and *Cut* .

path
The route used to locate files; the storage location of a file. A fully qualified path lists the drive identifier, directory name, subdirectory name (if any), and file name with the associated extension.

PDD
Physical device driver.

peeking
An action taken by any thread in the process that owns the queue to examine queue elements without removing them.

pel
(1) The smallest area of a display screen capable of being addressed and switched between visible and invisible states. Synonym for *display point* , *pixel* , and *picture element* . (2) (D of C) Picture element.

persistence
The quality of an entity such as a part, link, or object, that allows it to span separate document launches and transport to different computers. For example, a part unloaded to persistent storage is typically written to a hard disk.

persistent object
An object whose instance data and state are preserved between system shutdown and system startup.

persistent reference
A number, stored somewhere within a storage unit, that refers to another storage unit in the same document. Persistent references permit complex runtime object relationships to be stored externally, and later reconstructed.

physical device driver (PDD)
A system interface that handles hardware interrupts and supports a set of input and output functions.

pick

To select part of a displayed object using the pointer.

pickup
To add an object or set of objects to the pickup set.

pickup and drop
A drag operation that does not require the direct manipulation button to be pressed for the duration of the drag.

pickup set
The set of objects that have been picked up as part of a pickup and drop operation.

picture chain
See se*gment chain* .

picture element
(1) Synonym for *pel* . (2) (D of C) In computer graphics, the smallest element of a display surface that can be independently assigned color and intensity. (T) . (3) The area of the finest detail that can be reproduced effectively on the recording medium.

PID
Process identification.

pipe
(1) A named or unnamed buffer used to pass data between processes. A process reads from or writes to a pipe as if the pipe were a standard-input or standard-output file. See also *named pipe* and *unnamed pip*e . (2) (D of C) To direct data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (|).

pixel
(1) Synonym for *pel* . (2) (D of C) Picture element.

platform
The operating system environment in which a program runs. For example, OpenDoc is implemented on the Macintosh, Windows, and OS/2 platforms.

platform-normal coordinates
The native coordinate system for a particular platform. OpenDoc performs all layout and drawing in platform-normal coordinates; to convert from another coordinate system to platform- normal coordinates requires application of a bias transform.

plotter
An output unit that directly produces a hardcopy record of data on a removable medium, in the form of a two-dimensional graphic representation. (T)

PM
Presentation Manager.

pointer
(1) The symbol displayed on the screen that is moved by a pointing device, such as a *mou*se . The pointer is used to point at items that users can select. Contrast with *cursor* . (2) A data element that indicates the location of another data element. (T)

pointing device
In Advanced Common User Access architecture, an instrument, such as a mouse, trackball, or joystick, used to move a pointer on the screen.

pointings
Pairs of x-y coordinates produced by an operator defining positions on a screen with a pointing device, such as a *mou*se .

polyfillet
A curve based on a sequence of lines. The curve is tangential to the end points of the first and last lines, and tangential also to the midpoints of all other lines. See also *fillet* .

polygon
One or more closed figures that can be drawn filled, outlined, or filled and outlined.

polyline
A sequence of adjoining lines.

polymorphism
The ability to have different implementations of the same method for two or more classes of objects.

pop
To retrieve an item from a last-in-first-out stack of items. Contrast with *push* .

pop-up menu
A menu that lists the actions that a user can perform on an object. The contents of the pop-up menu can vary depending on the context, or state, of the object.

pop-up window
(1) A window that appears on top of another window in a dialog. Each pop-up window must be completed before returning to the underlying window. (2) (D of C) In Advanced Common User Access architecture, a movable window, fixed in size, in which a user provides information required by an application so that it can continue to process a user request.

port
(1) A connection between the CPU and main memory or a device (such as a terminal) for transferring data. (2) A socket on the back panel of a computer where you plug in a cable for connection to a network or a peripheral device.

port name
A unique identifier for a particular application within a computer. The port name contains a name string, a type string, and a script code. An application can specify any number of port names for a single port so long as each name is unique. See also *port* .

position code
A parameter (to a storage unit's Focus method) with which you specify the desired property or value to access.

preferences
The mechanism through which the user assigns a part editor to a given part kind.

preferred editor
The part editor that last edited a part, or for whom the part's data was just translated. If a part's preferred editor is not present, OpenDoc attempts to bind the part to the user s default editor for kind or default editor for category.

preferred kind
The part kind that a part specifies as its highest-fidelity, preferred format for editing. It is the part kind stored as the first value in the contents property of the part's storage unit, unless the storage unit also contains a property of type kODPropPreferredKind specifying another value as the preferred kind.

preferred representation
A property of a view controlling the default representation for parts contained inside it: either frame or icon. A part pasted or dragged into the part will change its representation to the containing view preferred representation. The user may override this on a part-by-part basis. For document parts the default view setting is frame; for desktop parts the setting is icon.

presentation
A particular style of display for a part's content-for example, an outline or expanded style for text, or a wire-frame or solid style for graphic objects. A part can have multiple presentations, each with its own rendering, layout, and user-interface behavior. See also *view type* .

presentation drivers
Special purpose I/O routines that handle field device-independent I/O requests from the PM and its applications.

Presentation Manager (PM)
The interface of the OS/2 operating system that presents, in windows a graphics-based interface to applications and files installed and running under the OS/2 operating system.

presentation page
The coordinate space in which a picture is assembled for display.

presentation space (PS)
(1) Contains the device-independent definition of a picture. (2) (D of C) The display space on a display device.

primary window
In Common User Access architecture, the window in which the main interaction between the user and the application takes place. In a multiprogramming environment, each application starts in its own primary window. The primary window remains for the duration of the application, although the panel displayed will change as the user's dialog moves forward. See also se*condary window* .

primitive
In computer graphics, one of several simple functions for drawing on the screen, including, for example, the rectangle, line, ellipse, polygon, and so on.

primitive attribute
A specifiable characteristic of a graphic primitive. See *graphics attribute*s .

print job
The result of sending a document or picture to be printed.

privilege level
A protection level imposed by the hardware architecture of the IBM personal computer. There are four privilege levels (number 0 through 3). Only certain types of programs are allowed to execute at each privilege level. See also *IOPL code segment* .

procedure call
In programming languages, a language construct for invoking - --- ü execution of a procedure.

process
An instance of an executing application and the resources it is using.

program

A sequence of instructions that a computer can interpret and execute.

program details
Information about a program that is specified in the *Program Manager* window and is used when the program is started.

program group
In the Presentation Manager, several programs that can be acted upon as a single entity.

program name
The full file specification of a program. Contrast with *program title* .

program title
The name of a program as it is listed in the *Program Manager* window. Contrast with *program name* .

promise
A specification of data to be transferred at a future time. If a data transfer involves a very large amount of data, the source part can choose to put out a promise instead of actually writing the data to a storage unit.

prompt
A displayed symbol or message that requests input from the user or gives operational information; for example, on the display screen of an IBM personal computer, the DOS A> prompt. The user must respond to the prompt in order to proceed.

properties notebook
The dialog box that displays allowing users to view a part's properties and to change any user-settable properties. It also allows access to the part's editor properties.

property
(1) In the OpenDoc storage subsystem, a component of a storage unit. A property defines a kind of information (such as 'name' or 'contents') and contains one or more data streams, called *value*s , that consist of information of that kind. Properties in a stored part are accessible without the assistance of a part editor. See also *display property, and user property* . (2) An OSA event object that defines some characteristic of another OSA event. object, such as its font or point size, that can be uniquely identified by a constant. The definition of each object class in the OSA Event Registry: Standard Suites lists the constants and class IDs for properties of OSA event objects belonging to that object class. For example, the constants pName and pBounds identify the name and boundary properties of OSA event objects that belong to the object class cWindow. The pName property of a specific window is defined by an OSA event object of object class cProperty, such as the word 'MyWindow,' which defines the name of the window. An OSA event object can contain only one of each of its properties, whereas it can contain many elements of the same element class. See also *OSA event object* , *container* , *element cla*sses .

pull-down
(1) An *action bar* extension that displays a list of choices available for a selected action bar choice. After users select an action bar choice, the pull-down appears with the list of choices. Additional *pop-up window*s may appear from pull-down choices to further extend the actions available to users. (2) (D of C) In Common User Access architecture, pertaining to a choice in an action bar pull-down.

push
To add an item to a last-in-first-out stack of items. Contrast with *pop* .

push button
In Advanced Common User Access architecture, a rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected.

putback
To remove an object or set of objects from the lazy drag set. This has the effect of undoing the pickup operation for those objects

putdown
To drop the objects in the lazy drag set on the target object.

Glossary - Q

queue
(1) A linked list of elements waiting to be processed in FIFO order. For example, a queue may be a list of print jobs waiting to be printed. (2) (D of C) A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed.

queued device context
A logical description of a data destination (for example, a printer or plotter) where the output is to go through the spooler. See also *device context* .

Glossary - R

radio button
(1) A control window, shaped like a round button on the screen, that can be in a checked or unchecked state. It is used to select a single item from a list. Contrast with *check box* . (2) In Advanced Common User Access architecture, a circle with text beside it. Radio buttons are combined to show a user a fixed set of choices from which only one can be selected. The circle is partially filled when a choice is selected.

raster
(1) In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. (T) (2) The coordinate grid that divides the display area of a display device. (A)

read-only file
A file that can be read from but not written to.

real mode
A method of program operation that does not limit or prevent access to any instructions or areas of storage. The operating system loads the entire program into storage and gives the program access to all system resources. Contrast with *protect mode* .

realize
To cause the system to ensure, wherever possible, that the physical color table of a device is set to the closest possible match in the logical color table.

recordable
A level of scripting support of a part. A recordable part allows the user to automatically convert user actions into scripts attached to the part. Compare with *scriptable, tinkerable* .

reference phrase help
In Common User Access architecture, highlighted words or phrases within help information that a user selects to get additional information.

refresh
To update a window, with changed information, to its current status.

region
A clipping boundary in device space.

registry
A dictionary that lists executable code modules and associated data by which they can be selected. Examples of a registry are *part registry* and *scripting component registry* .

remote file system
A file-system driver that gains access to a remote system without a block device driver.

remove
For a frame, to permanently delete it from its document, as well as from memory. Compare with close.

resource
The means of providing extra information used in the definition of a window. A resource can contain definitions of fonts, templates, accelerators, and mnemonics; the definitions are held in a resource file.

resource file
A file containing information used in the definition of a window. Definitions can be of fonts, templates, accelerators, and mnemonics.

restore
To return a window to its original size or position following a sizing or moving action.

retained graphics
Graphic primitives that are remembered by the Presentation Manager interface after they have been drawn. Contrast with *nonretained graphics* .

reverse video
(1) A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background. (2) In Basic Common User Access architecture, a screen emphasis feature that interchanges the foreground and background colors of an item.

revert
To return a draft to the state it had just after its last save.

REXX Language

Restructured Extended Executor. A procedural language that provides batch language functions along with structured programming constructs such as loops; conditional testing and subroutines.

RGB
(1) Color coding in which the brightness of the additive primary colors of light, red, green, and blue, are specified as three distinct values of white light. (2) Pertaining to a color display that accepts signals representing red, green, and blue.

roman
Relating to a type style with upright characters.

root facet
The facet that displays the root frame in a document window.

root frame
The frame in which the root part of a document is displayed. The root frame shape is the same as the content area of the document window.

root Part
The top-level part in a window opened from a user accessible object found in the files system, for example, a document that resides in an OS/2 folder. Any part can be a root part. The root part establishes the basic editing behavior of the document. Note that a root part need not be an OpenDoc part. It could be an application that embeds OpenDoc parts. In this case it effectively replaces the OpenDoc document shell.

root segment
In a hierarchical database, the highest segment in the tree structure.

Glossary - S

SAA
Systems Application Architecture.

save
To write all the data of all parts of a document (draft) to persistent storage.

SBCS
Single-byte character set.

screen
In Basic Common User Access architecture, the physical surface of a display device upon which information is shown to a user.

screen device context
A logical description of a data destination that is a particular window on the screen. See also *device context* .

script
A sequence of written instructions that, when executed by a script interpreter, are converted to semantic events that manipulate parts.

script editor
An application that allows users to record, edit, save, and execute scripts; for example, the Script Editor application.

script file
A file in which a script is stored. A script file can be a compiled script file, a script application file, or a script text file.

scriptable
A level of scripting support of a part. A scriptable part is able to accept semantic events for its publicly published content objects and operations. Compare with *tinkerable* and *recordable*.

scripting
Writing and executing scripts to control the behavior of multiple applications.

scroll bar
In Advanced Common User Access architecture, a part of a window, associated with a scrollable area, that a user interacts with to see information that is not currently allows visible.

scrollable entry field
An entry field larger than the visible field.

scrollable selection field
A selection field that contains more choices than are visible.

scrolling
Moving a display image vertically or horizontally in a manner such that new data appears at one edge, as existing data disappears at the opposite edge.

secondary window
A window that contains information that is dependent on information in a primary window and is used to supplement the interaction in the primary window.

sector
On disk or diskette storage, an addressable subdivision of a track used to record one block of a program or data.

segment
See *graphics segment* .

segment attributes
Attributes that apply to the segment as an entity, as opposed to the individual primitives within the segment. For example, the visibility or detectability of a segment.

segment chain
All segments in a graphics presentation space that are defined with the 'chained' attribute. Synonym for *picture chain* .

segment priority
The order in which segments are drawn.

segment store
An area in a normal graphics presentation space where retained graphics segments are stored.

select
(1) To mark or choose an item. Note that *select* means to mark or type in a choice on the screen; *enter* means to send all selected choices to the computer for processing. (2) In OpenDoc, to designate as the focus of subsequent editing operations. If the user selects an embedded part, that part's frame border takes on an appearance that designates it as selected. The embedded part's container is activated.

select button
The button on a pointing device, such as a mouse, that is pressed to select a menu choice. Also known as button 1.

selection cursor
In Advanced Common User Access architecture, a visual indication that a user has selected a choice. It is represented by outlining the choice with a dotted box. See also *text cursor* .

selection field
(1) In Advanced Common User Access architecture, a set of related choices. See also *entry field* . (2) In Basic Common User Access architecture, an area of a panel that cannot be scrolled and contains a fixed number of choices.

selection focus
The location of editing activity. The part whose frame has the selection focus is the active part, and has the selection or insertion point. See also *keystroke focus* .

semantic event
A message sent to a part or one of its content elements. Semantic events pertain directly to the part's content model and can have meaning independent of the part's display context. For example, semantic events could direct a part to get, set, or delete data. Compare with *user event* . See also *Open Scripting Architecture* .

semantic interface
A set of OpenDoc objects that provides an interface to allow parts to receive messages (semantic events) from other parts, in the same document or in other documents.

semantics
The relationships between symbols and their meanings.

semaphore
An object used by applications for signalling purposes and for controlling access to serially reusable resources.

separator
In Advanced Common User Access architecture, a line or color boundary that provides a visual distinction between two adjacent areas.

sequence number
A number that defines the position of a frame in its frame group.

serial dialog box
See *modal dialog box* .

serialization
The consecutive ordering of items.

serialize

To ensure that one or more events occur in a specified sequence.

serially reusable resource (SRR)
A logical resource or object that can be accessed by only one task at a time.

server application
An application that responds to OSA events requesting a service or information sent by client applications or scripting components (for example, by printing a list of files, checking the spelling of a list of words, or performing a numeric calculation). OSA event servers and clients can reside on the same local computer.

service
An OpenDoc component that, unlike a part editor, is not primarily concerned with editing and displaying parts. Instead, it provides a service to parts or documents, using the OpenDoc extension mechanism. Spelling checkers or database-access tools, for example, can be implemented as services.

session
(1) A routing mechanism for user interaction via the console; a complete environment that determines how an application runs and how users interact with the application. OS/2 can manage more than one session at a time, and more than one process can run in a session. Each session has its own set of environment variables that determine where OS/2 looks for dynamic-link libraries and other important files. (2) (D of C) In the OS/2 operating system, one instance of a started program or command prompt. Each session is separate from all other sessions that might be running on the computer. The operating system is responsible for coordinating the resources that each session uses, such as computer memory, allocation of processor time, and windows on the screen. (3) A logical connection between two entities (such as an OS/2 program and a database server) that facilitates the transmission of information between the two entities. An application has the option to accept or reject a session request. Authentication of the requesting user may be required before a session can commence. See also *authentication* , *message block* , *port* .

session ID
A number that uniquely identifies a session.

settings extension
An OpenDoc extension class that you can use to implement a Properties notebook.

shadow
An object that refers to another object. A shadow is not a copy of another object, but is another representation of the object.

shadow box
The area on the screen that follows mouse movements and shows what shape the window will take if the mouse button is released.

shape
A description of a geometric area of a drawing canvas.

shared data
Data that is used by two or more programs.

shared memory
In the OS/2 operating system, a segment that can be used by more than one program.

shared resource
A facility used by multiple parts. Examples of shared resources are the menu focus, selection focus, keystroke focus, and serial ports. See also *arbitrator* .

shear
In computer graphics, the forward or backward slant of a graphics symbol or string of such symbols relative to a line perpendicular to the baseline of the symbol.

shell
(1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens, and communicate them to the operating system. (2) Software that allows a kernel program to run under different operating-system environments.

show as frame
Displays a part as a frame. The space for the frame is obtained from the containing parts window or frame; it does not create a new window.

show as icon
Displays a part as an icon.

shutdown
The process of ending operation of a system or a subsystem, following a defined procedure.

sibling
A frame or facet at the same level of embedding as another frame or facet within the same containing frame or facet. Sibling frames and facets are z-ordered to allow for overlapping.

sibling processes
Child processes that have the same parent process.

sibling windows
Child windows that have the same parent window.

simple list
A list of like values; for example, a list of user names. Contrast with *mixed list* .

simple part
A part that cannot itself contain embedded parts. Compare c*ontainer part* .

single-byte character set (SBCS)
A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set* .

slider box
In Advanced Common User Access architecture: a part of the scroll bar that shows the position and size of the visible information in a window relative to the total amount of information available. Also known as *thumb mark* .

small icon view type.
A view type in which a part is represented by a 16 by 16 pixel bit map image. Other possible view types for displaying a part include large icon, thumbnail, and frame.

SOM
See *System Object Model* .

source content
The content at the source of a link. It is copied into the link and then into the destination content.

source data
Statements in a scripting language that constitute an uncompiled script.

source file
A file that contains source statements for items such as high-level language programs and data description specifications.

source frame
(1) An embedded frame whose part that has been opened up into its own part window. (2) The frame to which other synchronized frames are attached.

source part
A part that contains information that is displayed in another part through a link. Compare with *destination part* .

source statement
A statement written in a programming language.

spin button
In Advanced Common User Access architecture, a type of entry field that shows a scrollable ring of choices from which a user can select a choice. After the last choice is displayed, the first choice is displayed again. A user can also type a choice from the scrollable ring into the entry field without interacting with the spin button.

spline
A sequence of one or more Bézier curves.

split-frame view
A display technique for windows or frames, in which two or more facets of a frame display different scrolled portions of a part's content.

spooler
A program that intercepts the data going to printer devices and writes it to disk. The data is printed or plotted when it is complete and the required device is available. The spooler prevents output from different sources from being intermixed.

stack
A list constructed and maintained so that the next data element to be retrieved is the most recently stored. This method is characterized as last-in-first-out (LIFO).

standard window
A collection of window elements that form a panel. The standard window can include one or more of the following window elements: sizing borders, system menu icon, title bar, maximize/minimize/restore icons, action bar and pull-downs, scroll bars, and client area.

static canvas
A drawing canvas that cannot be changed after it has been rendered, such as a printer page. Compare with *dynamic canvas* .

static control
The means by which the application presents descriptive information (for example, headings and descriptors) to the user. The user cannot change this information.

storage system
The OpenDoc mechanism for providing persistent storage for documents and parts. The storage system object must provide unique identifiers for parts as well as cross-document links. It stores parts as a set of standard properties plus type-specific content data.

style
See *window style* .

subdirectory
In an IBM personal computer, a file referred to in a root directory that contains the names of other files stored on the diskette or fixed disk.

subsystem
A broad subdivision of the interface and capabilities of OpenDoc, involving one or more protocols (for example, OpenDoc subsystems include shell, storage, drawing, user events, and semantic events).

swapping
(1) A process that interchanges the contents of an area of real storage with the contents of an area in auxiliary storage. (I) (A) (2) In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage. (3) The process of temporarily removing an active job from main storage, saving it on disk, and processing another job in the area of main storage formerly occupied by the first job.

switch
(1) In CUA usage, to move the cursor from one point of interest to another; for example, to move from one screen or window to another or from a place within a displayed image to another place on the same displayed image. (2) In a computer program, a conditional instruction and an indicator to be interrogated by that instruction. (3) A device or programming technique for making a selection, for example, a toggle, a conditional jump.

switch list
See *Task List* .

symbolic identifier
A text string that equates to an integer value in an include file, which is used to identify a programming object.

symbols
In Information Presentation Facility, a document element used to produce characters that cannot be entered from the keyboard.

synchronized frames
Separate frames that display the same representation of the same part, and should therefore be updated together. In general, if an embedded part has two or more editable display frames of the same presentation, those frames (and all their embedded frames) should be synchronized.

synchronous
Pertaining to two or more processes that depend upon the occurrence of specific events such as common timing signals. (T) See also *asynchronous* .

System Menu
In the Presentation Manager, the pull-down in the top left corner of a window that allows it to be moved and sized with the keyboard.

System Object Model (SOM)
A mechanism for language-neutral, object-oriented programming.

system queue
The master queue for all pointer device or keyboard events.

system result handler
A result handler that is available to all applications that use the system. Compare with application result handler.

system-defined messages
Messages that control the operations of applications and provides input an other information for applications to process.

Systems Application Architecture (SAA)
A set of IBM software interfaces, conventions, and protocols that provide a framework for designing and developing applications that are consistent across systems.

Glossary - T

table tags
In Information Presentation Facility, a document element that formats text in an arrangement of rows and columns.

tag
(1) One or more characters attached to a set of data that contain information about the set, including its identification. (I) (A) (2) In Generalized Markup Language markup, a name for a type of document or document element that is entered in the source document to identify it.

target object
An object to which the user is transferring information.

Task List
In the Presentation Manager, the list of programs that are active. The list can be used to switch to a program and to stop programs.

template
A part that has the Templates property set. The newly created part is of the same type as the template and can contain new information generated during creation. The default drag operation of a template is Create. This allows documents containing boiler-plate to be easily used to create new documents. OpenDoc encourages developers to deliver their parts as templates.

text
Characters or symbols.

text cursor
A symbol displayed in an entry field that indicates where typed input will appear.

text window
Also known as the VIO window.

text-windowed application
The environment in which the operating system performs advanced-video input and output operations.

thumb mark
The portion of the scroll bar that describes the range and properties of the data that is currently visible in a window. Also known as a *slider box* .

thumbnail view type
A view type in which a part is represented by a large (64-by-64 pixels) bit map image that is typically a miniature representation of the layout of the part content. Other possible view types for displaying a part include large icon, small icon, and frame.

tilde
A mark used to denote the character that is to be used as a mnemonic when selecting text items within a menu.

timer tick
See *clock tick* .

tinkerable
A level of scripting support of a part. A tinkerable part allows the user to customize it, changing its behavior during virtually any user action. Compare with *scriptable* and *recordable* .

title bar
In Advanced Common User Access architecture, the area at the top of each window that contains the window title and system menu icon. When appropriate, it also contains the minimize, maximize, and restore icons. Contrast with *panel title* .

TLB
Translation lookaside buffer.

token
A short, codified representation of a string. The session object creates tokens for ISO strings. In OSA events for OpenDoc, a special descriptor structure that a part uses to identify one or more content objects within itself.

token disposal function
An object callback function that disposes of a token.

top-level part
The part that is visually topmost within a window. It might not be a root part. For example, when opened into its own window an embedded part is a top-level part in that window but it is not the root part of the document it is embedded within.

transform
(1) The action of modifying a picture by scaling, shearing, reflecting, rotating, or translating. (2) The object that performs or defines such a modification; also referred to as a *transformation* .

translation
The conversion of one type of data to another type of data. Specifically, the conversion of data of one part kind to data of another part kind.

Tree
In the Presentation Manager, the window in the *File Manager* that shows the organization of drives and directories.

truncate
(1) To terminate a computational process in accordance with some rule (A) (2) To remove the beginning or ending elements of a string. (3) To drop data that cannot be printed or displayed in the line width specified or available. (4) To shorten a field or statement to a specified length.

TSR
Terminate-and-stay-resident.

Glossary - U

undo
To rescind a command, negating its results. OpenDoc provides the ability to undo events by utilizing a command history.

unnamed pipe
A circular buffer, created in memory, used by related processes to communicate with one another. Contrast with *named pipe* .

unordered list
In Information Presentation Facility, a vertical arrangement of items in a list, with each item in the list preceded by a special character or bullet.

update region
A system-provided area of dynamic storage containing one or more (not necessarily contiguous) rectangular areas of a window that are visually invalid or incorrect, and therefore are in need of repainting.

update ID
(1) In OpenDoc, a number used to identify a particular instance of Clipboard contents. (2) A number used to identify a particular instance of link source data.

used shape
A shape that describes the portion of a frame that a part actually uses for drawing; that is, the part of the frame that the containing part should not draw over.

user event
A message, sent to a part by the dispatcher, that pertains only to the state of the part's graphical user interface, not directly to its contents. User events include mouse clicks and keystrokes, and they deliver information about, among other things, window locations and scroll bar positions. Compare with *semantic event* .

user interface
Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

user-interface part
A part without content elements, representing a unit of a document's user interface. Buttons and dialog boxes, for example, can be user-interface parts.

user property
One of a set of user-accessible characteristics of a part or its frame. The user can modify some user properties, such as the name of a part; the user cannot modify some other user properties, such as part category. Each user property defined by OpenDoc is stored as a distinct property in the storage unit of the part or its frame.

utility program
(1) A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. (T) (2) A program designed to perform an everyday task such as copying data from one storage device to another. (A)

Glossary - V

validate
To mark a portion of a canvas (or facet, or frame) as no longer in need of redrawing. Compare invalidate.

VGA
Video graphics array.

view
A way of looking at an object's information.

viewer
See *part viewer* .

view type
The basic visual representation of a part. Supported view types include frame, icon, small icon, and thumbnail.

viewing pipeline
The series of transformations applied to a graphic object to map the object to the device on which it is to be presented.

viewing window
A clipping boundary that defines the visible part of model space.

VIO
Video Input/Output.

virtual memory (VM)
Synonymous with *virtual storage* .

virtual storage
(1) The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations. (I) (A) (2) Addressable space that is apparent to the user as the processor storage space, from which the instructions and the data are mapped into the processor storage locations. (3) Synonymous with *virtual memory* .

visible region
A window's presentation space, clipped to the boundary of the window and the boundaries of any overlying window.

volume
(1) A file-system driver that uses a block device driver for input and output operations to a local or remote device. (I) (2) A portion of data, together with its data carrier, that can be handled conveniently as a unit.

Glossary - W

weak persistent reference
A persistent reference that, when the storage unit containing the reference is cloned, is ignored; the referenced storage unit is not copied. Compare strong persistent reference.

wildcard character
Synonymous with *global file-name character* .

window
(1) A portion of a display surface in which display images pertaining to a particular application can be presented. Different applications can be displayed simultaneously in different windows. (A) (2) An area of the screen with visible boundaries within which information is displayed. A window can be smaller than or the same size as the screen. Windows can appear to overlap on the screen. (3) A division of a screen in which one of several programs being executed concurrently can display information.

window canvas
The canvas attached to the root facet of a window. Every window has a window canvas.

window class
The grouping of windows whose processing needs conform to the services provided by one window procedure.

window-content transform
The composite transform that converts from a part's content coordinates to its window coordinates.

window coordinates
A set of coordinates by which a window position or size is defined; measured in device units, or *pel*s .

window-frame transform
The composite transform that converts from a part's frame coordinates to its window coordinates.

window handle
Unique identifier of a window, generated by Presentation Manager when the window is created, and used by applications to direct messages to the window.

window procedure
Code that is activated in response to a message. The procedure controls the appearance and behavior of its associated windows.

window rectangle
The means by which the size and position of a window is described in relation to the desktop window.

window resource
A read-only data segment stored in the .EXE file of an application o the .DLL file of a dynamic link library.

window state
An object that lists the set of windows that are open at a given time. Part editors can alter the window state, and the window state can be persistently stored.

window style
The set of properties that influence how events related to a particular window will be processed.

window title
In Advanced Common User Access architecture, the area in the title bar that contains the name of the application and the OS/2 operating system file name, if applicable.

Workplace Shell
The OS/2 object-oriented, graphical user interface.

workstation
(1) A display screen together with attachments such as a keyboard, a local copy device, or a tablet. (2) (D of C) One or more programmable or nonprogrammable devices that allow a user to do work.

world coordinates
A device-independent Cartesian coordinate system used by the application program for specifying graphical input and output. (I) (A)

world-coordinate space
Coordinate space in which graphics are defined before transformations are applied.

wrapper
An object (or class) that exists to provide an object-oriented interface to a non-object- oriented or system-specific structure. The OpenDoc class ODWindow, for example, is a wrapper for a system-specific window structure.

WYSIWYG
What-You-See-Is-What-You-Get. A capability of a text editor to continually display pages exactly as they will be printed.

Glossary - X

There are no glossary terms for this starting letter.

Glossary - Y

There are no glossary terms for this starting letter.

Glossary - Z

z-order
The order in which sibling windows are presented. The topmost sibling window obscures any portion of the siblings that it overlaps; the same effect occurs down through the order of lower sibling windows.

z-ordering
The front-to-back ordering of sibling frames used to determine clipping and event handling when frames overlap.

zooming
The progressive scaling of an entire display image in order to give the visual impression of movement of all or part of a display group toward or away from an observer. (I) (A)

8.3 file-name format
A file-naming convention in which file names are limited to eight characters before and three characters after a single dot. Usually pronounced "eight-dot-three." See also *non-8.3 file-name format* .

(No title)

For implementation efficiency, parts may not be deleted until some time after the last frame (or icon) has been deleted, however the user should not be aware of this. The Save choice periodically removes these unnecessary parts.

(No title)

For example, a clock part might have such few actions that there would be no need to activate the part. The user could select it and use the Selected pulldown of its container to perform actions on it.

(No title)

The name "Document" is the default name for the menu pull-down. The top-level part editor should change the name to the category name of the part or some equivalent name. For example, if the developer had a drawing object, "Drawing" could replace "Document" as the name of the menu when that part is the active part. Each part has its own window to display when it is active.

(No title)

Mnemonics for Icon, Tree, Details, and Drafts are **I**, **T**, **D**, and **F**.

(No title)

OpenDoc uses the clipboard actions **Cut**, **Copy**, **Paste**, and **Create**. WorkPlace Shell does not use the clipboard, but uses **Copy**, **Move**, and **Create**

(No title)

Mnemonics to be used for a menu choice are underlined.

(No title)

If the **Create** New window setting in the OS/2 System object is set, a new window will be opened.

(No title)

In addition to **Ctrl+P** for Print, the parts should support the OS/2 Presentation Manager standard of **Shift+Print Screen**.

(No title)

The **Edit** menu shows the use of **Ctrl+X**,**C**,**V** for **Cut**, **Copy**, and **Paste**. Parts should also support the OS/2 Presentation Manager standard of **Shift+Insert** for paste; **Ctrl+Insert** for Copy; and **Shift+Delete** for **Cut**.

(No title)

The name frame could be replaced with the name of the view, for example Chart or Table.

(No title)

Pressing Alt+MB1 promotes selection *up* one level, causing the currently-active part to become selected, and its container to become active. Each repeated action of pressing Alt+MB1 should promote selection up one level.

(No title)

The appropriate emphasis borders are illustrated in Appendix B.

(No title)

Developers should never look for the raw mouse events; instead look for semantic level messages.

(No title)

Instead of using the word "Frame" developers could list the actual name of the view. For example, if a part had both Chart and Table views, instead of listing Frame in the list, the part could list Chart and Table.

(No title)

The selection button default in OS/2 is MB1.

(No title)

If both the target and source reside on the same removable media, the drag would still be a move.

(No title)

This is the way a template works in OS/2.

(No title)

It might be useful to retain previous versions of an editor since different versions have different bugs or one version does not support older versions of the document format.

(No title)

Previously known as the Parts palette.

(No title)

The user might be able to view the a part in a document in multiple frames. This provides some of the capabilities as linking, but it is not as general a solution.

(No title)

All means links merged with intrinsic content or those embedded in the active part. Links inside of other embedded parts would not be shown.

(No title)

This is guideline for developers, OpenDoc cannot guarantee that formatting will be preserved.

(No title)

Theoretically, OpenDoc could try to preserve some of these links, but the rules for when a link will and will not be broken are so complex that users cannot understand them.
